VŠB – Technická univerzita Ostrava

Fakulta elektrotechniky a informatiky

Katedra informatiky

Online predaj

Online sale

2013 Martin Boor

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Martin Boor**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Online prodej
Online Sale

Zásady pro vypracování:

Cílem práce je naprogramovat systém podpory prodeje výrobků přes internet. Tento systém bude mimo samotný prodej umožňovat uživatelům naplánování cesty k místu dodání s výpočtem předpokládaných nákladů na dopravu za pomocí online map. Součástí systému budou statistiky prodeje, nákladů spojených s prodejem a dopravou.
Student nastuduje problematiku tvorby informačních systémů, provede průzkum trhu s ohledem na využívání stávajících systémů. Student provede srovnání stávajících informačních systémů.

Práce bude obsahovat:
1. Popis technik vývoje informačního systému.
2. Průzkum stávajícího stavu, jaký freeware v dané oblasti je k dispozici, jeho výhody, nevýhody.
3. Analýzu, návrh a implementaci informačního systému. Tento systém bude umožňovat naplánování trasy dopravy k zákazníkovi, odhadované a reálné náklady na dopravu a samotný prodej.

Systém bude naprogramován v jazyce C#.

Seznam doporučené odborné literatury:

[1] Voříšek, Bruckner, Buchalcevová: Tvorba informačních systémů, GRADA, ISBN: 978-80-247-4153-6,
[2] Sodomka: Informační systémy, Computer Press, ISBN: 80-251-1200-4

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Mgr. Marek Menšík, Ph.D.**

Datum zadání: 16.11.2012
Datum odevzdání: 07.05.2013

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Čestné prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne a uviedol som všetky literárne zdroje a publikácie z ktorých som čerpal informácie.

V Ostrave 02.04.2013 .....

Poďakovanie

Týmto by som sa chcel poďakovať Mgr. Marekovi Menšíkovi, Ph.D. - môjmu odbornému konzultantovi, za jeho odborné konzultácie a cenné pripomienky a rady. Poďakovanie patrí aj mojej najbližšej rodine, ktorá ma počas môjho štúdia podporovala.

Abstrakt

Cieľom mojej bakalárskej práce je analyzovať, navrhnúť a implementovať informačný systém na podporu online predaja výrobkov cez internet. Okrem samotného predaja umožní užívateľom plánovanie ciest ku zákazníkom s výpočtom predpokladaných nákladov. Výsledný systém predstavuje unikátna webová aplikácia, postavená na platforme .NET. Aplikácia obsahuje rozhranie pre zákazníka, ako aj pre administrátora. Práca obsahuje aj teoretické spracovanie problematiky tvorby informačných systémov a prieskum súčasného stavu trhu s informačnými systémami pre elektronické obchody.

Kľúčové slová: informačný systém, e-shop, Google mapy, .NET, MVC, Entity Framework, LINQ

Abstract

The aim of my thesis is to analyze, design and implement an information system to support online sales of products over the Internet. Apart from the sale allows the user to plan trips to customers with the calculation of estimated costs. The resulting system is a unique web-based application built on the platform. NET. The application includes an interface for the customer as well as the administrator. Thesis includes theoretical treatment of problems of information systems and survey the current stage of market of information systems for e-commerce.

Keywords: information system, e-shop, Google maps, .NET, MVC, Entity Framework, LINQ

Zoznam použitých skratiek a symbolov

ASD	Adaptive software development
AJAX	Asynchronous JavaScript and XML
API	Aplication programming interface
CMMI	Capability Maturity Model Integration
CAPTCHA	CAPTure CHAracters
DFD	Data Flow Diagram
ERM	Entity relationship model
EUP	Enterprise unifield process
JS	Javascript
MSF	Microsoft Solution Framework
MVC	Model-view-controller
ORM	Object relational mapping
OSS	Open source software
PHP	Hypertext processor
RUP	Rational unifield process
SCRUM	agilná metodika vývoja softwaru
SW	Software
UML	Unified modeling language

Obsah

Zoznam tabuliek

Zoznam obrázkov

1 Úvod

Informačné systémy sú v posledných rokoch neoddeliteľnou súčasťou nášho sveta. Uľahčujú alebo celkom automatizujú vykonávanie procesov potrebných pre fungovanie podnikov, bánk ale aj verejných služieb. Služby informačných systémov využívame v dnešnej dobe každodenne, či sa jedná o využívanie privátneho firemného systému v zamestnaní alebo využitím služieb internet bankingu, sociálnych sietí či ide o výber hotovosti z bankomatu.

Rozvojom webových technológií a masovým nárastom užívateľov internetu, sa veľká časť obchodovania presunula na internet. Predaj výrobkov alebo služieb cez internetu prináša mnoho výhod pre zákazníka ako aj pre predajcu. Pre zákazníka je pohodlné ak si môže tovar zakúpiť z pohodlia svojho domova, zaplatiť ho online bez potreby hotovosti a nechať si ho doviezť priamo domov. Často produkty predávané na internete konkurujú produktom z kamenných obchodov hlavne svojou cenou. Nižšia cena produktov býva zapríčinená znížením nákladov práve presunutím podnikania z kamenných obchodov na internet. Často ale podniky rozširujú svoje pôsobenie na internet so zachovaním kamenných pobočiek.

Cieľom tejto práce je vytvoriť systém pre podporu predaja výrobkov na internete, s možnosťou jeho administrácie. Systém bude zahŕňať aj administračnú sekciu na správu produktov, objednávok, zákazníkov a zamestnancov, systém na plánovanie ciest ku zákazníkom a štatistiky spojené s predajom a nákladmi na chod obchodu.

1.1 Štruktúra bakalárskej práce

Práca je rozdelená do troch hlavných častí. *Techniky a metodiky vývoja informačných systémov, Prieskum súčasného stavu trhu,* a *Analýza, návrh a implementácia vlastného riešenia.*

Prvá časť sa venuje rozdeleniu informačných systémov, životným cyklom softwarového produktu a popisu techník a metodík vývoja informačných systémov.

V druhej kapitole je prieskum súčasného trhu s informačnými systémami, prehľad a porovnanie dostupného freeware a komerčného softwaru. Okrajovo sa venujem aj individuálnym riešeniam softwaru na mieru.

Tretia kapitola opisuje samotnú analýzu, návrh a implementáciu vlastného riešenia systému. Popisuje použité techniky a technológie a niektoré zaujímavé implementačné techniky. V podkapitole *Analýza* sa detailne venujem zberu a analýze požiadaviek na systém. *Návrh* sa zaoberá výberom konkrétnych technológií a postupov a obsahuje aj samotný návrh riešenia. Podkapitola *Implementácia* sa zaoberá konkrétnymi postupmi pri implementácií, ukážkami zdrojových kódov zložitejších algoritmov alebo zaujímavých implementačných postupov.

2 Informačný systém

2.1 Definícia

Podľa [1], môžeme informačný systém definovať ako skupinu navzájom poprepájaných komponent, ktoré majú za úlohu zabezpečiť vstup, spracovanie a uchovávanie dát. Ďalej umožňuje vykonávať akcie ktoré tieto dáta premieňajú na informačné produkty, ktoré môžu byť použité na podporu plánovania, kontroly, koordinácie, rozhodovania a iné činnosti v organizáciach.

Komponenty, ktoré tieto aktivity zabezpečujú, môžeme klasifikovať do piatich základných skupín: ľudia, hardware, software, komunikácia a dáta.

Ľudské zdroje zahŕňajú všetkých užívateľov systému, vývojárov, IS manažérov, technickú podporu a všetkých ľudí, ktorí prichádzajú so systémom do kontaktu.

Hardwarové zdroje pokrývajú všetku technickú výbavu ako počítače, servery a príslušenstvo k nim ako tlačiarne, skenery, faxy a pod.

Softwarové zdroje zahŕňajú programové vybavenie hardwaru, ktoré obsahuje programy, aplikácie, operačné systémy a iný software potrebný pre beh organizácie a jej IS.

Komunikačné zdroje zahŕňajú siete, hardware a software, ktorý je potrebný pre zabezpečenie komunikácie.

Dátové zdroje zastrešujú všetky dáta, ku ktorým má organizácia prístup, či sa jedná o počítačové databázy, alebo papierové dokumenty.

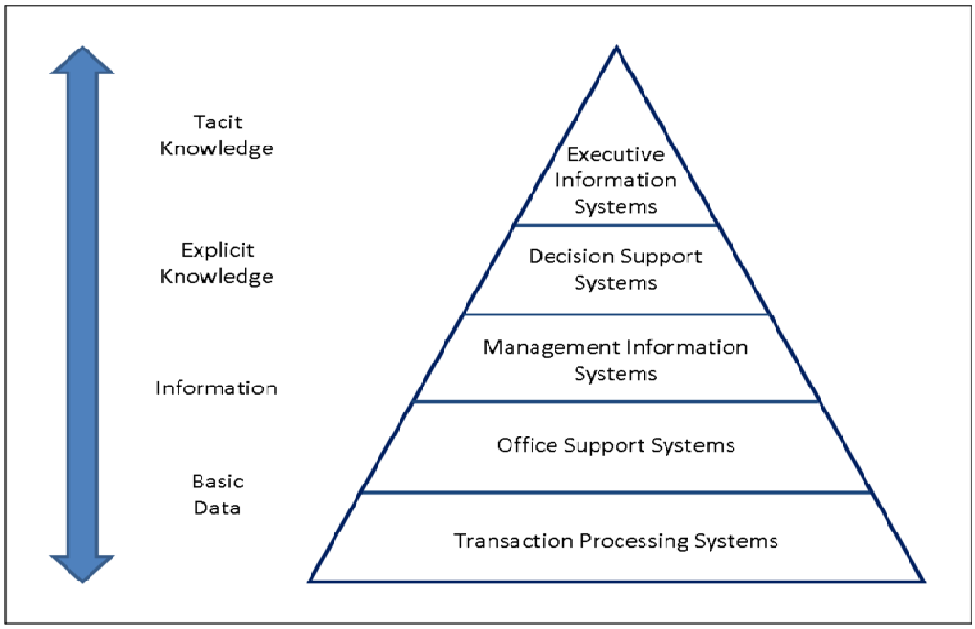
Komponenty informačného systému sa často veľmi podobajú komponentám biznis systému, preto môžeme informačný systém chápať ako súčasť biznis systému. V informačnom systéme sa kladie dôraz skôr na informácie o komponente (o človeku, stroji, materiály a pod.) ako na komponentu samotnú.

2.2 Rozdelenie informačných systémov

V tejto kapitole vychádzam z [3] a [4]. Rôzny užívatelia informačných systémov majú rôzne nároky na informácie s ktorými potrebujú pracovať. Vrcholový manažment potrebuje informácie potrebné na plánovanie biznisu, Stredný manažment potrebuje detailnejšie informácie na monitorovanie a riadenie rôznych aktivít spojených s výrobou alebo predajom a zamestnanci potrebujú informácie ktoré im pomôžu pri plnení svojich každodenných úloh.

Informačné systémy môžeme rozdeliť do piatich hlavných kategórií podla zamerania (Obrázok 1):

- Systémy na spracovanie transakcií (TPS – Transaction Processing Systems)
- Kancelárske (OSS – Office Support Systems)
- Manažérske (MIS – Management Information Systems)
- Taktické (DSS – Decision Support Systems)
- Systémy pre výkonný manažment (EIS – Executive Information Systems)



Obr. : 1 Pyramídový model typov informačných systémov (Zdroj: [3])

Veľa systémov svojou funkcionalitou pokrýva viacero z týchto kategórií. Napríklad internetový obchod zväčša mimo základnej funkcie riadenia transakcií (Spracovanie objednávok, platieb, ai.), obsahuje aj nástroje pokrývajúce kategóriu manažérskych systémov (štatistiky, prehľady, reporty ai.).

Systémy na spracovanie transakcií

Ako už názov napovedá, TPS sú určené na spracovávanie rutinných transakcií a operácií. TPS sa snažia väčšinu procesov zautomatizovať a tým zvýšiť efektivitu práce a znížiť náklady.

Dobrým príkladom TPS sú napr.: Fakturačné systémy, Systémy pre výpočty miezd, daňových platieb, Systémy riadenia zásob a zdrojov, Elektronické obchody, bankové služby ai.

Kancelárske systémy

OSS – Office Support Systems sú systémy, ktoré sa snažia zvýšiť produktivitu zamestnancov, ktorý pracujú s dátami a informáciami. Jedná sa najmä o rôzne kancelárske softwarové balíky (Microsoft Office, Libre Office), komunikačné nástroje, nástroje na prácu z domova apod.

Manažérske systémy

Manažérske informačné systémy (MIS) poskytujú informácie strednému manažmentu, ktoré pomáhajú pri monitoringu, riadení a plánovaní byznys akcií. Často zhromažďujú dáta z TPS a sumarizujú ich do sérií manažérskych reportov.

Taktické systémy

DSS - Decision Support Systems sú špecificky navrhnuté systémy, ktoré pomáhajú manažmentu pri rozhodovaní. Jedná sa hlavne o situácie, kedy nie je jasné aké výsledky alebo dôsledky rozhodnutie resp. zmena prinesie. DSS poskytujú nástroje a techniky, ktoré pomáhajú zbierať relevantné informácie a analyzovať možnosti a alternatívy. Výsledkom bývajú tzv. „What-if“ modely.

Systémy pre výkonný manažment

Executive Support Systems (ESS) pomáhajú výkonnému manažmentu robiť strategické rozhodnutia. Zbierajú, analyzujú a sumarizujú kľúčové interné a externé informácie používané v byznyse. Snažia sa podať komplexný prehľad o stave všetkých kľúčových byznys aktivít a pomocou analytických a modelovacích nástrojov ako napr. „What-if“ analýz pomôcť vedeniu pri dôležitých strategických rozhodnutiach.

2.3 Elektronické podnikanie a elektronické obchodovanie

Vzhľadom na zameranie tejto bakalárskej práce sa budem zvlášť venovať pojmom elektronické podnikanie (E-business) a elektronické obchodovanie (E-commerce).

2.3.1 Elektronické podnikanie

Elektronické podnikanie zahŕňa všetky aktivity spojené s prevádzkou a podporou obchodovania na internete. Podľa [1], je tvorená nasledovnými vrstvami:

- Internetová infraštruktúra – Spoločnosti zabezpečujúce hardware, software a iné príslušenstvo dôležité k behu internetu (ISP, sieťové spoločnosti, výrobcovia počítačov a serverov, apod.)
- Infraštruktúra internetových aplikácií – Spoločnosti dodávajúce software ktorý realizuje internetové transakcie, web-dizajnérske spoločnosti, konzultantské spoločnosti apod.
- Internetový sprostredkovatelia – Spoločnosti ktoré spájajú nakupujúcich a predávajúcich napr. vytváraním a zabezpečovaním tzv. marketplaces (v preklade: obchodných miest), kde sa môže uskutočniť obchodná transakcia medzi predávajúcim a kupujúcim (Cestovné kancelárie, rezervačné služby apod.)
- Internetové obchodovanie – Spoločnosti, ktoré predávajú produkty a služby zákazníkom alebo iným spoločnostiam. Príkladom sú on-line obchody, predplatitelské služby apod.

Hlavné výhody elektronického podnikania sú v znížení nákladov, zvýšením efektivity a rozšírením spektra pôsobnosti. Automatizovaním značného množstva administratívnych úloh spojených s objednávkami, zásobovaním a doručovaním tovaru a služieb, sa výrazne zníži cena celého biznis procesu resp. cena konečného výrobku alebo služby.

2.3.2 Elektronické obchodovanie

Hlavná aktivita spojená s elektronickým podnikaním je podľa [1], elektronické obchodovanie. Hlavnou úlohou elektronického obchodovania je zabezpečiť vykonanie byznys transakcií ako sú nakupovanie a predávanie tovaru a služieb využitím informačných technológií. Elektronické obchodovanie môže zahŕňať širokú škálu ďalších aktivít spojených s popredajnou podporou, logistikou, riadením skladových zásob apod.

Elektronické obchodovanie môžeme rozdeliť do 5 základných kategórií:

- Business-to-business (B2B) – Obchodné transakcie medzi spoločnosťami. Približne 80% elektronického obchodovania je tohto typu.
- Bussiness-to-consumer (B2C) – Spoločnosti predávajú produkty priamo zákazníkom
- Bussiness-to-government (B2G) – Obchodné transakcie medzi spoločnosťou a organizáciami verejného sektoru
- Consumer-to-consumer (C2C) – Obchodné transakcie medzi dvoma zákazníkmi, kde jeden je v roli predávajúceho a druhý v roli nakupujúceho. Najlepším príkladom takéhoto systému sú rôzne aukčné servery.

3 Vývoj informačných systémov

Vývoj informačných systémov je veľmi náročná a komplexná disciplína. V [2] je uvedené, že z prieskumu realizovanom spoločnosťou Standish Group vyplýva, že v roku 1994 bolo úplne úspešných len 16% softwarových produktov a v roku 2009 to bolo už 32%. Je vidieť že za 15 rokov sa úspešnosť softwarových projektov zvýšila, za čo môže vývoj a aplikácia rôznych metodík, štandardov a noriem, ale 32% je stále pomerne malá úspešnosť, preto sa treba v tomto smere neustále zlepšovať.

„*Zvýšiť úspešnosť softwarových projektov sa snažia ako tradičné prístupy tak agilné prístupy. Každý z nich však k tomu pristupuje na základe iných predpokladov. Tradičné prístupy považujú tvorbu IS za definovaný proces, ktorý je možno presne popísať, podľa tohto popisu opakovane realizovať a zlepšovať. Agilné prístupy naopak vychádzajú z presvedčenia, že proces tvorby IS je empirický proces, ktorý nemá zmysel popisovať, ale je potreba ho monitorovať a prispôsobovať realite.*“ [2]

Ako som uviedol vyššie, tradičné prístupy považujú vývoj IS za definovaný proces ktorý možno opísať. Preto sa pri tradičných prístupoch môžeme stretnúť s rôznymi referenčnými modelmi procesov, modelmi posudzovania zrelosti a spôsobilosti procesov, modelmi životného cyklu softwaru, a tradičnými metodikami.

3.1 Referenčné modely procesov a posudzovania procesov

3.1.1 Model CMMI

V tejto kapitole som vychádzam z [2], [5] a [6]. CMMI je skratka pre Capability Maturity Model Integration – Integračný model zrelosti, ktorý v roku 2000 vytvoril Software Engineering Institute (SEI) na Carnegie Mellon University. CMMI je kolekcia osvedčených postupov na podporu zvyšovania úrovne vývoja, služieb a akvizícií v podniku. Model CMMI môžeme nazvať ako „Referenčný model“, pretože osvedčené postupy je možné použiť ako referencie pre porovnávanie organizácií na základe zrelosti procesov v nich. Tento model je stále v aktívnom vývoji a momentálne je k dispozícií vo verzií 1.3 a obsahuje tieto modely:

- **CMMI-ACQ** pre akvizície – poskytuje návody na zlepšenie procesov v organizáciách ktoré sa zaoberajú zaobstarávaním softwaru, systémov alebo hardwaru tretích strán.
- **CMMI-DEV** pre vývoj – poskytuje návody na zlepšenie procesov v organizáciách ktoré vyvíjajú software, systémy aj hardware.

- **CMMI-SVC** pre služby – poskytuje návody na zlepšenie procesov pri poskytovaní služieb.

Podľa [2], môžeme CMMI-DEV definovať ako referenčný model procesov, ktorý pokrýva celý životný cyklus softwarového produktu. Proces je v CMMI modeli definovaný ako sada vzájomne previazaných činností, ktoré transformujú vstupy na výstupy pre dosiahnutie definovaného účelu. Model CMMI-DEV definuje 22 procesných oblastí. Procesná oblasť (Process Area) je skupina navzájom spojených praktík, ktoré pokiaľ implementujeme kolektívne, naplní množinu cieľov, ktoré sú podstatné pre zlepšenie danej oblasti. Model CMMI-DEV podporuje dva spôsoby zlepšovania procesov – kontinuálny a stupňovitý. V kontexte kontinuálneho zlepšovania sa používa pojem úroveň spôsobilosti (Capability Level), ktorá je definovaná ako dosiahnutie zlepšenia procesov v určitej procesnej oblasti. Stupňovitá reprezentácia pracuje s úrovňou zrelosti (Maturity Level), ktorá predstavuje stupeň zlepšenia procesov v rámci preddefinovanej množiny procesných oblastí, v ktorej sú dosiahnuté všetky definované ciele.

Úroveň	**Kontinuálna reprezentácia** **Úroveň spôsobilosti**	**Stupňovitá reprezentácia** **Úroveň zrelosti**
0	Neúplný (Incomplete)	
1	Vykonávaný (Performed)	Úvodná (Initial)
2	Riadený (Managed)	Riadená (Managed)
3	Definovaný (Defined)	Definovaná (Defined)
4		Kvantitatívne Riadená (Quantitavely Managed)
5		Optimalizovaná (Optimizing)

Tab. 1 Prehľad úrovní spôsobilosti a zrelosti v CMMI-DEV zdroj: [2]

Kontinuálna reprezentácia pomáha zlepšovať procesy len v niektorej procesnej oblasti.

- **Úroveň 0 – Neúplný proces** sa nevykonáva vôbec alebo iba čiastočne. Na tejto úrovni nie sú splnené niektoré alebo žiadne ciele procesu.
- **Úroveň 1 – Vykonávaný proces** transformuje vstupy na výstupné pracovné produkty. Ciele procesu a procesnej oblasti sú naplnené ale proces nie je súčasťou politiky spoločnosti. Tým pádom nie je možné zaistiť jeho konzistentné vykonávanie.
- **Úroveň 2 – Riadený proces** je vykonávaný proces, ktorý je plánovaný a vykonávaný v súlade s podnikovou politikou. Proces je monitorovaný, kontrolovaný a vyhodnocovaný.
- **Úroveň 3 -Definovaný proces** je riadený proces ktorý je prispôsobovaný špecifickým podmienkam. Je presne definovaný jeho účel, vstupy, výstupy, kritéria, aktivity, úlohy, opatrenia a kroky pre overenie procesu.

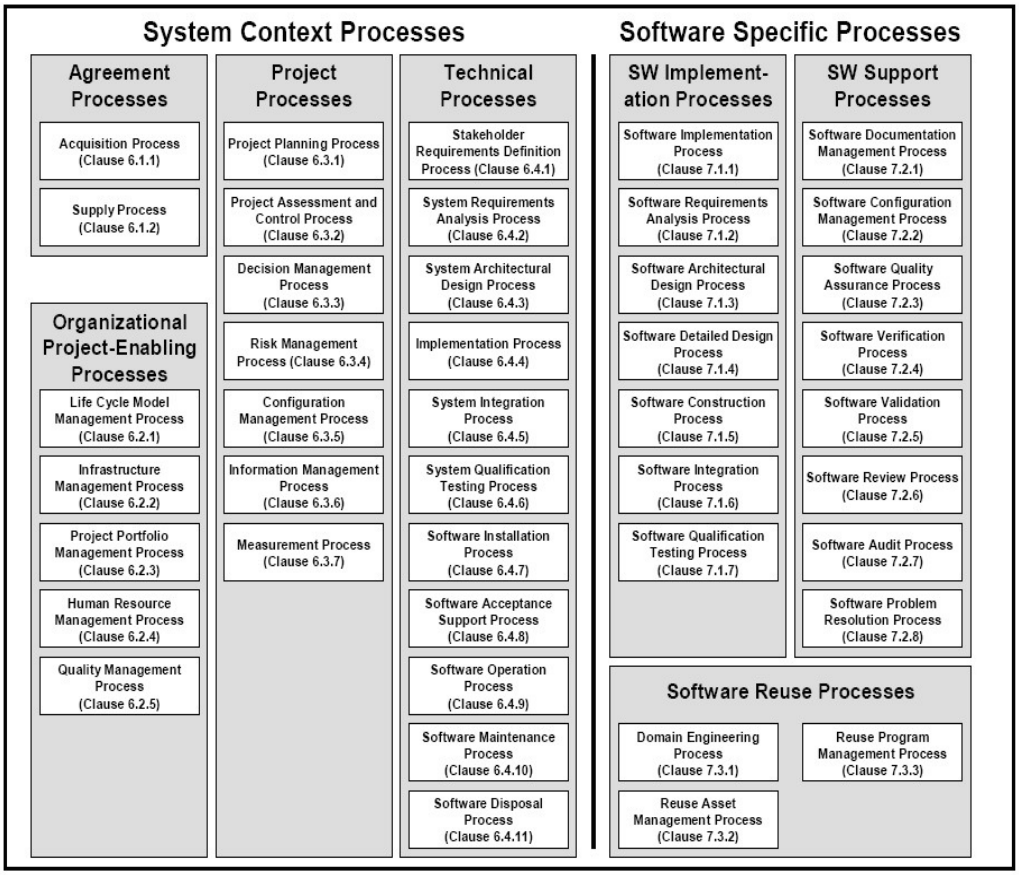
„*Stupňovitá reprezentácia definuje pre každú úroveň zrelosti (Maturity Level) množinu procesných oblastí, ktoré je treba zaviesť.*“ [2]

- **Úroveň 1 – Úvodná** zahŕňa organizácie, ktoré nemajú softwarové procesy riadené a často sú náhodné a chaotické. Úspech takejto organizácie závisí hlavne na kompetenciách a výkone zamestnancov ako na použití osvedčených postupov.
- **Úroveň 2 – Riadená** zahŕňa organizácie, ktoré využívajú procesy na riadenie projektov, procesy sú súčasťou podnikovej politiky, sú plánované a realizované na základe definovaného popisu, sú monitorované a kontrolované. Patria sem tieto procesné oblasti: *Project Planning, Project Monitoring and Control, Supplier Agreement Management, Requirements Management, Measurment and Analysis, Process and Product Quality Assurance, Configuration Management*
- **Úroveň 3** - **Definovaná** zahŕňa organizácie ktoré dosiahli úroveň zrelosti 2. Proces je súčasťou podnikovej politiky, je popísaný v štandardoch, procedúrach, nástrojoch a metódach. Proces je naviac prispôsobovaný jednotlivým projektom. Patria sem tieto procesné oblasti: *Requirements Development, Technical Solution, Product Integration, Verification, Validation, Decision Analysis and Resolution, Organizational Process Focus, Organizational Process Definition, Organisational Training, Integrated Project Management, Risk Management*
- **Úroveň 4 – Kvantitatívne riadená** zahŕňa organizácie ktoré splnili úrovne zrelosti 2 a 3. Na riadenie procesov sa využívajú štatistické a kvantitatívne techniky, sú definované dielčie podprocesy, ktoré sa merajú a vyhodnocujú na základe čoho je možné predpovedať výkon procesov. Patria sem procesné oblasti: *Organisational Process Performance* a *Quantitative Process Management*
- **Úroveň 5 – Optimalizovaná** zahŕňa organizácie, ktoré dosiahli všetkých cieľov úrovní 2,3 a 4. Procesy v organizácií sa neustále skúmajú a zlepšujú. Patria sem tieto procesné oblasti: *Organizational Performance Management* a *Causal Analysis and Resolution*

3.1.2 Medzinárodné normy ISO/IEC pre softwarové procesy

ISO/IEC 12207 Procesy v životnom cykle softwaru

Je norma vydaná v roku 1995 a naposledy revidovaná v roku 2008. Definuje sadu procesov, činností a úloh, dôležitých pri dodávke, vývoji, prevádzkovaní, údržbe a odstraňovaní softwarového produktu. Definuje 34 procesov rozdelených do 7 procesných skupín (Obrázok 2). Pri každom procese jasne definuje názov, účel, vstupy, výstupy a aktivity spojené s vykonávaním procesu.



Obr. 2 Procesy životného cyklu SW podľa ISO/IEC 12207(zdroj: [7])

Norma ISO/IEC 15504 Information technology – Process assesment

Táto norma definuje postupy pri posudzovaní úrovne spôsobilosti procesov tvorby IS a stanovuje minimálne požiadavky zaisťujúce konzistentnosť a opakovateľnosť posudzovania. Môže byť použitá buď s cieľom dokázať určitú úroveň spôsobilosti procesov za účelom certifikácie alebo môže byť použitá ako východisko pre zlepšovanie úrovne procesov. [2]

Táto norma má rovnaký účel ako vyššie spomínaný model CMMI-DEV. Umožňuje posudzovať úroveň spôsobilosti jednotlivých procesov ale aj celej organizácie. Norma definuje 6 úrovní spôsobilosti:

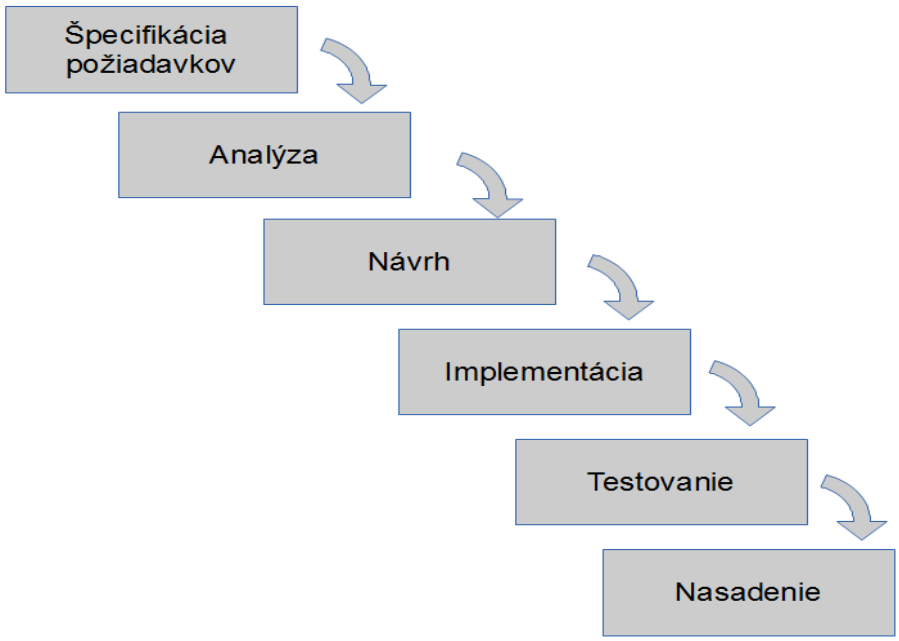
- Úroveň 0 – **Nekompletný proces**
- Úroveň 1 – **Vykonávaný proces**
- Úroveň 2 – **Riadený proces**
- Úroveň 3 – **Zavedený proces**
- Úroveň 4 – **Predvídateľný proces**
- Úroveň 5 - **Optimalizovaný proces**

3.2 Modely životného cyklu SW

„*Životný cyklus systému je časový úsek, ktorý začína úmyslom vytvoriť systém a končí, keď sa systém prestane používať*.“ [2] Môžeme ho charakterizovať ako súbor procesov a aktivít často organizovaných do fáz.

3.2.1 Vodopádový model

Podľa [1] a [2] sa tento model rozšíril najmä v 70. a 80 rokoch. Inšpirovaný postupmi v priemysle rozdelil vývoj SW do postupne realizovaných fáz. Celý proces tvorí 6 fáz: *Špecifikácia požiadaviek, Analýza, Návrh, Implementácia, Testovanie, Nasadenie*. Svoj názov si získal skutočnosťou, že jednotlivé fázy sú vykonávané postupne a pripomínajú vodopád. viď. obrázok 3.



Obr. 3 Vodopádový model životného cyklu SW

V dobe svojho vzniku predstavoval vodopádový model významný pokrok. Vďaka rozdeleniu celého procesu na samostatné fázy umožnil systematický a opakovateľný postup vývoja. Tento model sa v praxi stále používa hlavne pri projektoch, pri ktorých je možné v prvej fáze špecifikácie požiadaviek, definovať všetky požiadavky a požiadavky sa počas vývoja zásadne nemenia. Problémy s týmto modelom nastávajú v situáciách kedy nie sú na začiatku cyklu známe všetky požiadavky alebo v prípadoch kedy sa požiadavky často menia počas vývoja. Ďalšími problémami je dlhý časový úsek od zadania projektu po jeho konečnú implementáciu a nasadenie, a spolupráca so zákazníkom, ktorý je do celého procesu tvorby IS

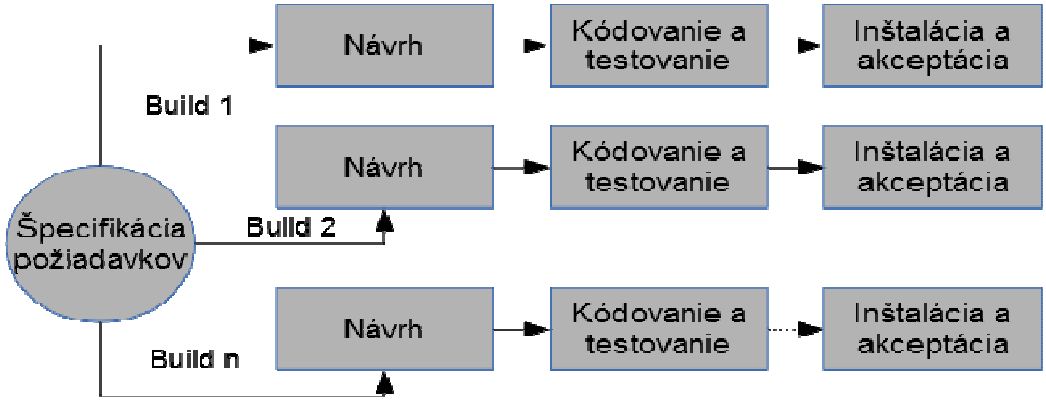
zapojený len na začiatku a na konci.

3.2.2 Modely pre iteratívny vývoj

Iteratívny vývoj [2] odstraňuje nevýhody vodopádového modelu tým, že rozkladá celý projekt na menšie subprojekty – iterácie. Každá iterácia prechádza všetkými fázami vývoja. V grafickej reprezentácií by každú jednu iteráciu predstavoval jeden malý vodopád. Výsledkom každej iterácie je spustiteľná, otestovaná a integrovaná aplikácia. Rozdelením projektu na menšie časti sa znižujú riziká pri vývoji SW, pretože v každej iterácií je možné otestovať funkcionalitu a správnosť systému. Iteratívny vývoj reprezentujú dva základné modely životného cyklu: Inkrementálny a Evolučný.

Inkrementálny model

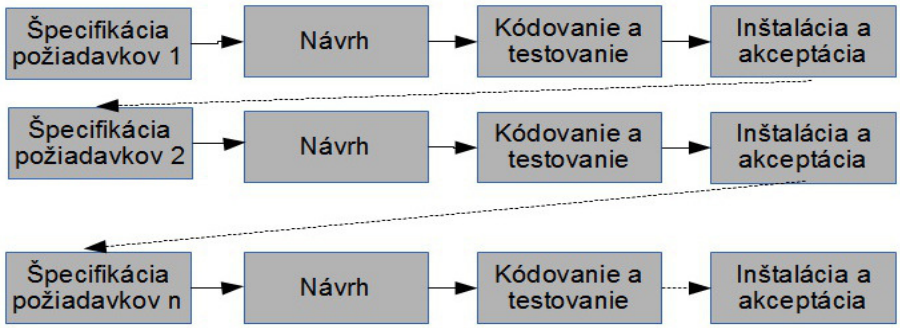
Pri tomto modeli sa všetky požiadavky definujú na začiatku projektu. Projekt je potom rozdelený na samostatne realizovateľné časti (prírastky), ktoré prechádzajú všetkými fázami vývoja.



Obr. 4 Inkrementálny model

Evolučný model

Evolučný model je takmer totožný s inkrementálnym modelom, líši sa od neho v tom, že všetky požiadavky nie sú definované na začiatku projektu ale definujú sa až na začiatku každej iterácie.

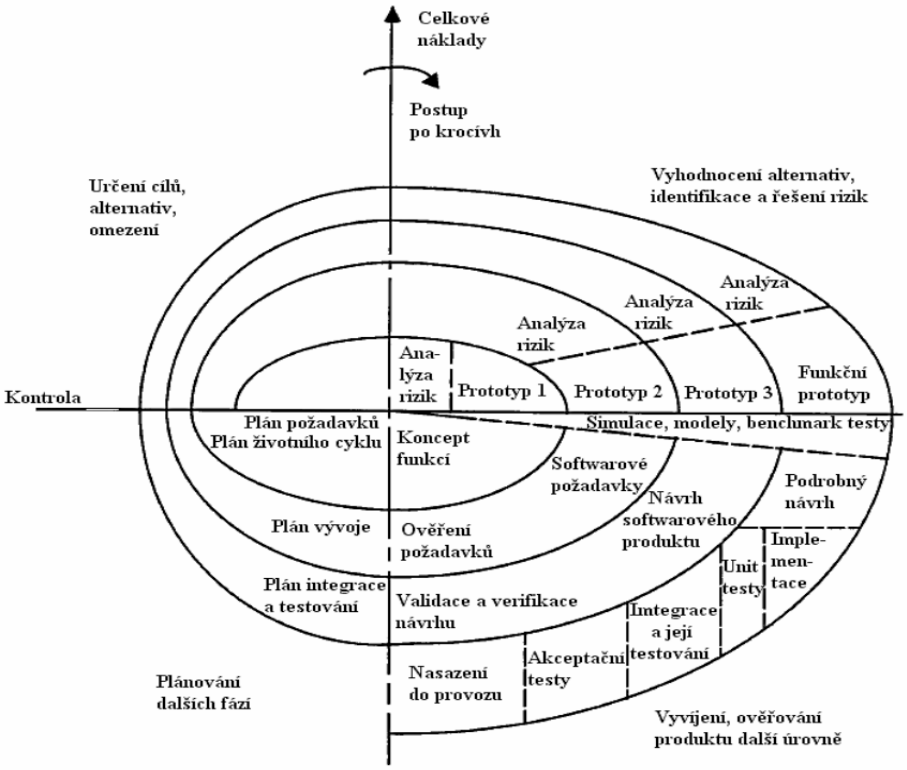


Obr. 5 Evolučný model

Špirálový model

Tento model, definovaný Barrym Boehmom, pokrýva najväčšie nedostatky vodopádového modelu a patrí do skupiny tzv. prístupov riadených rizikami. Model je založený na iteratívnom prístupe vývoja a pozostáva zo štyroch hlavných častí (obr. 6):

- Určenie cieľov, alternatív a obmedzení
- Vyhodnotenie alternatív, identifikácia a riešenie rizík
- Vývoj a verifikácia ďalšej úrovne produktu
- Plánovanie nasledujúcich fáz



Obr. 6 Špirálový model, zdroj: [15]

3.3 Metodiky tvorby IS

Ako som už spomenul na začiatku tejto kapitoly, proces tvorby informačného systému a všeobecne softwarového produktu je veľmi zložitý a komplexný. Metodika je súhrn metód, techník a postupov pre realizáciu určitej úlohy, v tomto prípade tvorby IS. Rôzne metodiky pokrývajú rôzne fázy životného cyklu. Niektoré sa zameriavajú len na jednotlivé projekty, iné na riadenie procesov v celej organizácii

V súčasnosti prevládajú v oblasti metodík dva hlavné prúdy. Tradičné resp. rigorózne (ťažké) metodiky a agilné (ľahké) metodiky. Niektoré metodiky kombinujú obidva tieto prístupy, keď k rigoróznym metódam a technikám pridávajú aj agilné metódy a postupy.

3.3.1 Metodika Rational Unified Process

„*Metodika Rational Unified Process je založená na tzv. najlepších praktikách softwarového vývoja*“ [2]. Patrí do rigoróznych metodík a charakterizujú ju tieto vlastnosti:

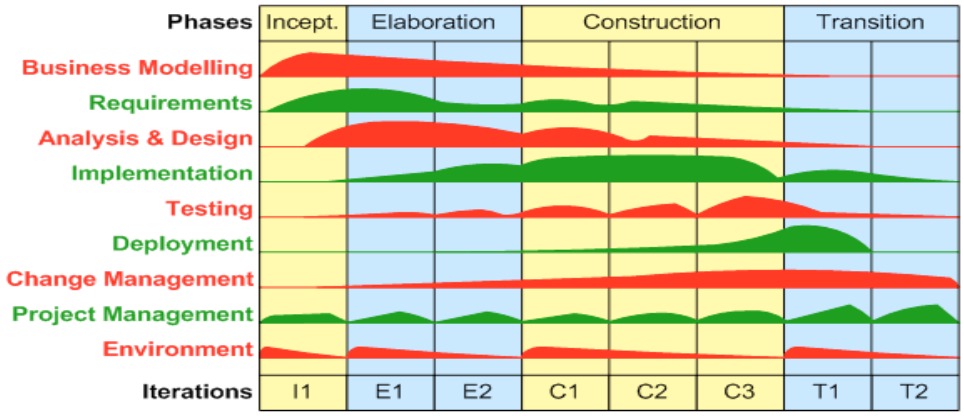
- iteratívny vývoj
- riadenie požiadaviek
- použitie komponentovej architektúry
- vizuálne modelovanie
- kontrola kvality SW
- riadenie zmien

Proces vývoja je zachytený na obrázku 7. a je popísaný v dvoch dimenziách. Prvá dimenzia reprezentovaná horizontálnou osou (x), predstavuje dynamický pohľad na proces a je vyjadrená pomocou cyklov, fáz, iterácií a míľnikov. Druhú dimenziu predstavuje vertikálna os (y) a vyjadruje statický pohľad na proces. Obsahuje popis činností, artefaktov, pracovníkov a pracovných tokov rozdelených do 9 disciplín, predstavujúcich základné zoskupenie hlavných činností, ktoré metodika RUP definuje. Graf vo vnútri obrázka reprezentuje podiel práce v každej disciplíne v jednotlivých fázach projektu.

Vývojový cyklus pozostáva zo štyroch fáz [9]:

- **Počiatočná fáza (Inception)** – V tejto fáze sa definujú ciele a požiadavky projektu, náklady a riziká spojené s projektom. Projekt sa rozdeľuje na iterácie a vytvára sa harmonogram projektu (iterácií). V tejto fáze sa ukáže, či je možné na základe definovaných požiadaviek, technológií a rozpočtu projekt realizovať.

- **Elaboračná fáza (Elaboration)** – V tejto fáze by sa mala definovať architektúra vyvíjaného systému a vytvoriť prvý prototyp architektúry ktorý overí jej funkčnosť a správnosť architektúry. Definujú sa komponenty systému ktoré je možno znovu použiť.
- **Konštrukčná fáza (Contruction)** – Táto fáza sa zaoberá návrhom, realizáciou a testovaním systému. Ak je to možné, využíva sa paralelný vývoj.
- **Nasadenie (Transition)** – táto fáza zaisťuje nasadenie hotového systému pre produkčné použitie. Táto fáza zahŕňa aj aktivity spojené so školením a podporou užívateľov.



Obr. 7 Rational Unified Process – fázy a disciplíny. Zdroj: [8]

V každej fáze prebieha jedna alebo viacero iterácií ktorej sa venuje všetkých 9 disciplín. Výsledkom každej iterácie je funkčný systém s nejakou pridanou funkcionalitou.

Metodika RUP je dodávaná s nástrojom Rational Method Composer, ktorý slúži na správu metodiky. Obsahuje nástroje na správu webového obsahu, konfiguráciu metodického obsahu a procesov a veľa ďalších nástrojov umožňujúcu maximálne využitie tejto metodiky.

3.3.2 Metodika Enterprise Unified Process

Táto metodika je v podstate rozšírením metodiky RUP. Kým RUP je svojim zameraním projektová metodika, Enterprise Unified Process EUP zachováva vyššie uvedené vlastnosti a rozširuje túto metodiku na úroveň celej organizácie, pridaním troch ďalších disciplín:

- **Infrastructure Management** – zahŕňa procesy realizované projektmi
- **Production** – zastrešuje procesy spojené s prevádzkou a údržbou systému.
- **Retirement** – popisuje procesy a činnosti potrebné na odstránenie produktu z prevádzky .

Metodika EUP prekonáva obmedzenie metodiky RUP, ktorá je zameraná len jednotlivý projekt a je globálnou metodikou zameranou na vývoj IS na úrovni celého podniku. [6]

3.3.3 Metodika Microsoft Solutions Framework

Microsoft Solutions Framework (MSF) je framework pre vývoj softwaru, ktorý pokrýva metodiku pre konkrétne organizácie ako aj konkrétny projekt. Posledná verzia MSF podporuje agilné prístupy a tímový vývoj softwaru. [2]

MSF definuje dva modely:

- **MSF Team model** – definuje, ako organizovať tímy pracovníkov a ich činnosti tak, aby bol projekt úspešný
- **MSF Process model** - definuje procesy, ich ciele a výstupy, ktoré je nutné aplikovať pre úspešné vytvorenie softwarového produktu

Tímový model

Tímový model rozdeľuje pracovníkov do siedmich skupín. (tabuľka 2)

Rola/Skupina	Popis
Projektový manažment	Riadenie projektu, riešenie obmedzení projektu
Vývoj	Návrh a implementácia riešenia podľa špecifikácií
Testovanie	Testovanie systému podľa špecifikácií a reportovanie výsledkov testov
Skúsenosti užívateľov	Zvyšovanie efektívnosti užívateľov
Uvoľňovanie/Nasadenie	Hladké nasadzovanie verzií a finálneho systému
Produktový manažment	Uspokojovanie potrieb zákazníka

Tab. 2 Pracovné skupiny tímového modelu MSF [10]

V každej skupine sú definované role ako napr. Analytik, Produktový manažér, Projektový manažér, Vývojár, Tester a jednotlivé aktivity, ktorými výstupmi sú pracovné produkty. Pre podporu riadenia projektu a komunikácie medzi tímami sa generujú rôzne hlásenia na projektovom portáli. [10]

Procesný model

Tento model je založený na kombinácií vodopádového a špirálového modelu životného cyklu. Vyznačuje sa krátkymi iteráciami, ktoré končia nasadením novej verzie s novou funkcionalitou, ktorá prináša zákazníkovi pridanú hodnotu. Procesný model je navrhnutý tak, aby umožňoval zmeny požiadaviek aj v rámci prebiehajúceho vývoja.

Model sa skladá z piatich základných fáz: Tvorba vízie (Envisioning), Plánovanie (Planning), Vývoj (Developing), Stabilizácia (Stabilization), Nasadenie (Deploying). Každá fáza končí takzvaným kontrolným bodom v ktorom sa kontroluje, či boli dosiahnuté všetky ciele danej fáze a rozhoduje sa o ďalšom pokračovaní projektu.

Obr. 8 MSF procesný model [10]

V pozadí celého modelu prebieha ešte fáza Governance ktorá zahŕňa aktivity spojené s riadením projektu. Ide o: Riadenie rizík, Príprava tímu a Riadenie projektu. [2]

Metodika je dodávaná s vývojovým prostredím Visual Studio Team System, podporujúcim celú metodiku.

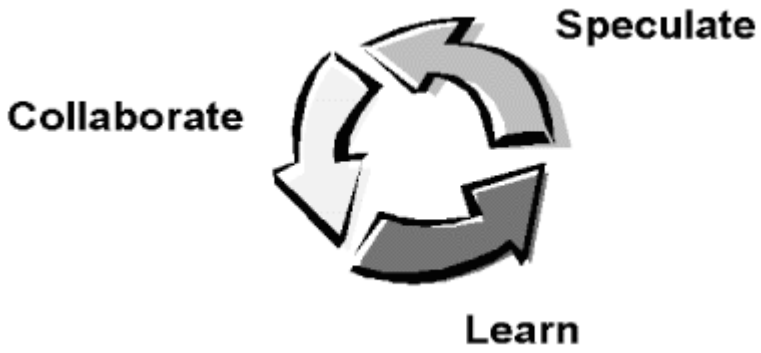
3.3.4 Adaptive Software Development

Metodika Adaptive Software Development (ASD) podľa [15] predstavuje filozofické zázemie agilných metodík. Metodika nahrádza klasický statický životný cyklus Plánovanie-Návrh-Tvorba novým dynamickým modelom Špekulácie-Spolupráca-Učenie.

Najväčším problémom klasického životného cyklu je fáza plánovanie, pretože po tejto fáze je problematickejšie zadávanie nových požiadaviek a inovácií. Preto dáva fáza špekulácie viacej priestoru pre zmeny, podporuje skúmanie a experimentovanie.

Dôležitou časťou tohto modelu je Spolupráca. Pri velkých projektoch je pri práci nevyhnutná spolupráca všetkých členov tímu, komunikácia a výmena informácií medzi tímami. Tými spoločne riešia ako technické problémy tak aj analýzu požiadavkou.

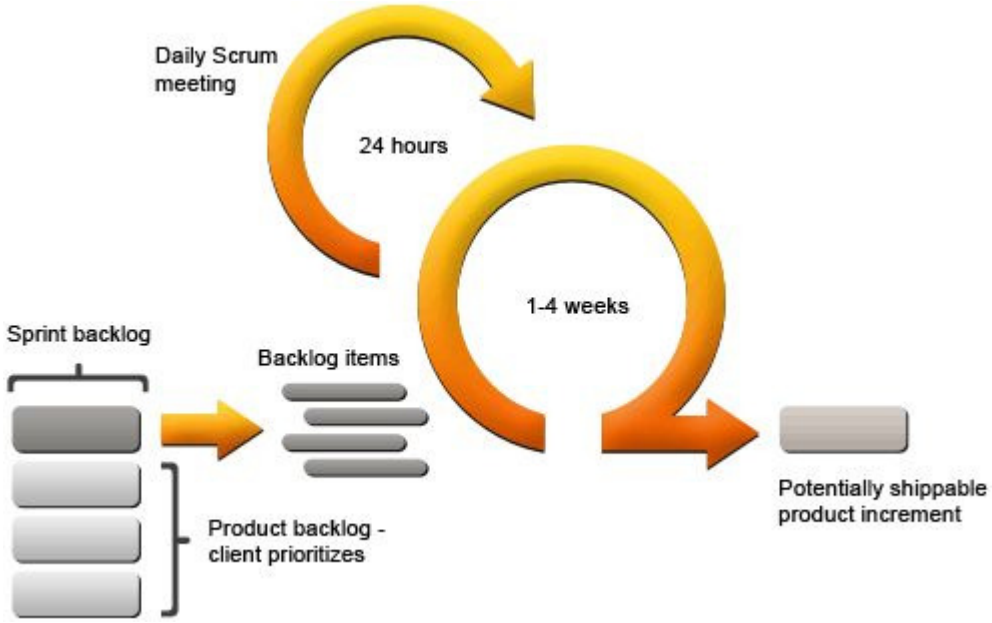
Vo fáze učenia sa hodnotí kvalita produktu zo strany zákazníka aj z technologického hľadiska, stav projektu, fungovanie tímov.



Obr. 9 Životný cyklus pre ASD zdroj [15]

3.3.5 Metodika SCRUM

Metodika SCRUM je najpoužívanejšou agilnou metodikov. Zameriava sa hlavne na riadenie projektu. Postup vývoja je ukázaný na obrázku 10.



Obr. 10 Postup vývoja podľa metodiky SCRUM. [16]

Ako popisuje autor v [17], v prvej fáze sa špecifikujú prvé funkčné požiadavky na produkt, rozdelenie projektu na menšie časti a plán dodávok.

V druhej fáze sú pridané aj nefunkčné požiadavky.

V tretej fáze jeden alebo viacero tímov implementuje funkcionalitu systému a v maximálne mesačných intervaloch dodáva a predvádza výsledok .kľúčovou praktikou SCRUM sú každodenné krátke porady (cca 15 minút) Scrum Meetings, ktoré slúžia na koordináciu práce a identifikáciu problémov. Riadi ju Scrum Master. Každý člen tímu odprezentuje, aké úlohy

dokončil od minulej porady, aké nové úlohy bude riešiť a aké vidí v riešení daného problému prekážky.

SCRUM definuje len tri role:

- **Product Owner** – spravuje zoznam požiadaviek (Product Backlog)
- **Team** – tvorí skupina ľudí s rôznou špecializáciou, ktorá sa sama riadi tak aby na konci každého Sprintu dodali fungujúci software.
- **Scrum Master** – zodpovedá za správne použitie metodiky a maximalizáciu jej využitia

4 Prieskum trhu s internetovými obchodmi

Obchodovanie na internete išlo v posledných pár rokoch veľmi dopredu. V súčasnosti je k dispozícií veľké množstvo hotových riešení elektronických obchodov.

Niektoré sú vydávané ako Open Source software, ktorý je možno ľubovoľné upravovať a prispôsobovať, prevádzkovať na vlastných zariadeniach a pod.

Iné riešenia elektronických obchodov sú poskytované alebo predávané ako služby, ktoré umožňujú zákazníkovi vytvorenie vlastného obchodu pomocou predpripravených šablón. Celú službu zabezpečuje a prevádzkuje jedna organizácia.

Poslednou skupinou sú elektronické obchody a systémy vytvorené na zákazku špecializovanou firmou presne podľa zákazníkových požiadaviek. Táto možnosť finančne najnáročnejšia.

4.1 Hotové riešenia

V tejto kapitole sa budem zaoberať hotovými riešeniami elektronických obchodov. V prvej podkapitole *Open Source Software* opíšem a porovnám niektoré voľne dostupné e-shopy. V druhej podkapitole s názvom *E-shop ako služba* popíšem niektoré platené riešenia elektronických obchodov.

4.1.1 Open Source Software

Open Source Software (OSS) – je druh softwaru ktorý je voľne dostupný aj so svojim zdrojovým kódom. Nazýva sa tiež otvorený software, nielen pre technickú dostupnosť kódu ale aj legálnu dostupnosť – licenciou. Táto licencia umožňuje pri dodržaní jej podmienok slobodné šírenie, modifikáciu a používanie softwaru.

Tieto licencie musia vyhovovať definíciám organizácie Open Source Initiative (OSI). Patria sem najmä tieto: GPLv3, LGPL , AGPL, OSLv3 BSD, MIT, Apache

Magento Comunity Edition

Tento Open Source projekt pod záštitou spoločnosti Magento je obľúbeným a často

používaným riešením pre elektronický obchod. Je vydávaný pod licenciou OSL 3.0, ktorá umožňuje jeho modifikáciu a šírenie. Umožňuje sťahovanie digitálneho obsahu, prispôsobovanie vzhľadu pomocou skinov, jednoduchú prácu s katalógom produktov, objednávkami, skladovými zásobami, štatistikami a WISIWIG editorom Pre užívateľa sú zaujímavé funkcie ako Newsletter, podpora tlače, vyhľadávanie produktov, hodnotenie produktov, Wish-list, darčekové poukazy, členské výhody, špeciálne ponuky ako aj rôzne možnosti platieb (Paypal, 2CheckOut, WorldPay, kreditné karty) alebo prepojenie so sociálnymi službami. [11]

Ako implementačný jazyk je použité PHP v kombinácii s databázou MySQL. Iné databázy nie je možné použiť.

Najnovšou verziou je ver 1.7.0.2 z dňa 5.6.2012.

nopCommerce

Riešenie od spoločnosti nopCommerce je vydávané pod licenciou nopCommerce Public License (GPL + dodatky o reklame) a ponúka bohatú škálu možností ako pre administrátora obchodu tak aj pre zákazníka. Má podobné vlastnosti a funkcie ako má vyššie uvedený systém. Pridáva k nim niektoré doplňujúce funkcie a nástroje ako napr. Možnosť prevádzkovania tzv. pobočiek, eliminácia robotov pomocou CAPTCHA. [12]

Je naprogramovaný v jazyku c# za použitia ASP.NET MVC Framework. Umožňuje prácu s databázami MySQL a MSSQL.

Momentálne je vo verzií 2.80 z dňa 5.1.2013

KonaKart

Systém od spoločnosti DS Data Systems UK Ltd je vydávaný pod licenciou LPGL. Okrem základnej práce s katalógom, objednávkami a akciami, ponúka SEO optimalizáciu, kontrolu prístupu zamestnancov.

Systém je naprogramovaný v Jave a umožňuje pracovať s veľkým množstvom najpoužívanejších databáz. Medzi podporované databázy patria: MySQL, PosgreSQL, MSSQL a Oracle. Veľkou výhodou použitia jazyka java je jeho multiplatformnosť. [13]

Posledná komunitná verzia je 6.5.1.0

Zen Cart

Tento projekt je pod záštitou spoločnosti Zen Ventures a je vydávaný pod licenciou GPL. Je to jednoduchý systém pre elektronické obchody. Jednoduchý je hlavne preto, lebo nepoužíva

niektoré bežné moderné technológie ako napr. Ajax a neobsahuje veľké množstvo funkcionality ako napr. import a export dát, interaktívne vyhľadávanie, prepojenie so sociálnymi službami. Funkcionalita sa dá dodatočne rozšíriť pomocou plug-inov, čo býva bežné pre každý podobný systém. [14]

Systém je naprogramovaný v PHP v kombinácií s databázou MySQL. Iné databázy nie sú podporované.

Momentálne je vo verzií 1.5.1

Kompletná tabuľka s porovnávanými hodnotami je v prílohe A.

4.1.2 E-shop ako služba

Systémy opísané vyššie sú voľne dostupné na stiahnutie. Pre zabezpečenie chodu je potreba nasadiť systém na webový server, nakonfigurovať databázu a vykonať iné pre bežného používateľa technicky náročnejšie úlohy. Okrem iného nie je garantovaná žiadna profesionálna technická podpora. V súčasnosti je možné zaobstarať si internetový obchod ako službu, ktorú zastrešuje a prevádzkuje spoločnosť zameraná na túto problematiku. Spoločnosti ponúkajú väčšinou svoje vlastné riešenia, v rôznych variáciách (konfiguráciách) a cenových reláciách. K službe patrí vytvorenie e-shopu podľa predpripravených šablón, samotný hosting, zákaznícku podporu a iné doplnkové funkcie.

V tejto kapitole popíšem a porovnám tri takéto služby, spoločností ktoré pôsobia na slovenskom a českom trhu.

Eshop-rychlo.sk

Túto službu tvorí kompletný systém pre vytvorenie a prevádzku internetového obchodu alebo webovej prezentácie. Obsahuje nástroje pre správu vzhľadu (250 voľne dostupných šablón + možnosť vlastného návrhu), nástroje pre správu obsahu (oznamy, bannery, stránky, apod.) a správu katalógu (produkty, kategórie, varianty, apod.) Všetky informácie pochádzajú z [18].

Je k dispozícií jeden základný balík (ktorý je možné rozšíriť) ktorého cena začína od 6,94 € + DPH / mesiac, a má tieto vlastnosti:

- vlastná doména
- 1GB priestoru na disku
- 5x E-mailová adresa
- fotogaléria
- komentáre

- ankety
- bannerová reklama
- skladová evidencia
- akčný tovar
- súvisiaci tovar
- fakturácia
- export do PDF
- on-line platby (karta, PayPal)
- SEO optimalizácia
- Technická podpora

Webaster.sk

Táto služba takisto zahŕňa systém pre správu obsahu ako aj systém pre správu e-shopu. Zaujímavou a veľmi užitočnou funkciou je tzv. ISWID (I see what i am doing) editor, ktorý umožňuje užívateľovi editovať celý vzhľad a obsah stránok bez znalosti programovania. Práca s ISWID je založená na Drag-and-Drop a umožňuje tak užívateľovi vytvoriť jedinečný vzhľad stránok alebo e-shopu. Všetky informácie pochádzajú z [19].

Balík zahŕňajúci CMS systém a E-shop je k dispozícii od 8,90 € bez DPH / mesiac a zahŕňa tieto funkcie:

- vlastná doména
- 5 GB priestoru na disku
- neobmedzený počet e-mailových adries
- live chat
- voľba jazyka stránok
- fotogaléria
- mailinglist
- varianty produktov
- editácia objednávok
- fakturácia

- skladové zásoby
- import / export produktov
- SEO optimalizácia

Systém je možné rozšíriť rôznymi doplnkovými službami za príplatok.

FastCentrik

Toto riešenie od spoločnosti NetDirect poskytuje systém pre prevádzku internetového obchodu, ktorý ponúka jednoduchú správu produktov a prispôsobenie vzhľadu pomocou šablón.

V najzákladnejšej verzií je k dispozícií od 11 € bez DPH a podľa [20] má tieto parametre:

- vlastná doména
- 1GB priestoru na disku
- neobmedzený počet e-mailových adries
- novinky
- skladová evidencia
- súvisiaci tovar
- zľavy
- registrácia do cenových porovnávačov
- pravidelná aktualizácia systému
- technická podpora

Svojou cenou je najzaujímavejší systém Eshop-rychlo.sk, ktorý ponúka slušnú sadu funkcií. Systém Webaster.sk ponúka tiež za pomerne nízku cenu veľa funkcií, z ktorých najužitočnejšou je ISWID editor. Aplikácia FastCentrik ponúka veľa možností rozšírenia systému pomocou modulov. Výhodou je bezplatná registrácia produktov do porovnávačov cien.

4.2 Riešenia na mieru

Ak je potrebné zaobstarať riešenie so špecifickými potrebami a vlastnosťami je možné využiť služieb softwarových spoločností, ktoré ponúkajú tvorbu softwaru od základov podľa špecifikácií zákazníka. Je ťažké porovnať ceny rôznych poskytovateľov takýchto služieb, pretože každé riešenie býva špecifické svojou náročnosťou, rozsahom a tým pádom aj cenou. Takáto možnosť je spravidla finančne najnáročnejšia, pretože vyžaduje individuálny prístup k riešeniu a je aj časovo náročnejšia.

5. Analýza návrh a implementácia vlastného riešenia

5.1 Zadanie a špecifikácia

Mojim cieľom je vyvinúť informačný systém na podporu predaja tovaru cez internet s administráciou a systémom pre plánovanie ciest ku zákazníkom. Riešenie by malo byť prispôsobené na podporu predaja a rozvozu produktov v pizzerií.

5.1.1 Základné požiadavky

Zo zadania vyplýva, že systém bude rozdelený do dvoch sekcií. Sekcia pre zákazníka a administračná sekcia pre predajcu.

Zákaznícka sekcia

Pre neprihláseného zákazníka bude prístupná len časť zákazníckej sekcie, ktorá bude umožňovať zákazníkovi prehliadanie tovaru s možnosťou vloženia produktu do košíka, registráciu nového zákazníka a prihlásenie registrovaného zákazníka.

Po prihlásení do systému, bude umožnené zákazníkovi pristúpiť k vytvoreniu záväznej objednávky z nákupného košíka, upravovať osobné údaje v profile a prezerať svoje objednávky.

Zhrnutie funkcií:

- Registrácia
- Prihlásenie
- Prehliadanie tovaru
- Nákupný košík
- Objednávka
- Úprava profilu

Administračná sekcia

Do administračnej sekcie budú mať prístup len zamestnanci, ktorých môžeme rozdeliť do troch kategórií: Predajca, Rozvozca a Administrátor. Tieto užívateľské role sa líšia

pridelenými prístupovými právami.

Predajca by mal mať kompletný prístup k objednávkam, čo zahŕňa ich vytváranie, editáciu a mazanie.

Rozvozca by mal mať plnú kontrolu nad plánovaním ciest. Jednotlivé body cesty sa budú vyvárať z objednávok ktoré sú pripravené na export. Systém plánovania ciest bude umožňovať rozvozcovi pridávať jednotlivé objednávky do plánovanej cesty a vypočítať a zobraziť trasu na mape. Po naplánovaní bude možné cestu uložiť do systému. Po uložení sa zmení aj status objednávky.

Administrátor má kompletný prístup ku všetkým vyššie uvedeným funkciám. Naviac môže pridávať, editovať a mazať kategórie, produkty, varianty produktov a zamestnancov.

Všetci zamestnanci môžu prezerať štatistiky a meniť svoje heslá.

Zhrnutie funkcií:

- Kategórie
- Produkty
- Varianty produktov
- Objednávky
- Plánovanie ciest
- Správa zamestnancov
- Štatistiky

5.1.2 Zoznam vstupov

- Objednávka – kontaktné informácie zákazníka, dátum, status objednávky, cena
- Status objednávky – názov
- Položka objednávky – názov produktu, cena, celková cena (súčet základnej ceny produktu a cien vybraných variant ako napr. veľkosť, farba), špeciálne prianie
- Varianta položky objednávky – názov varianty, cena varianty
- Produkt – názov, cena, váha, kategória, popis produktu
- Kategória produktu – názov, obrázok kategórie, obrázok
- Obrázok produktu – názov, URL
- Varianta produktu – názov, cena
- Cesta – vzdialenosť, čas, dátum, zamestnanec ktorý cestu uložil

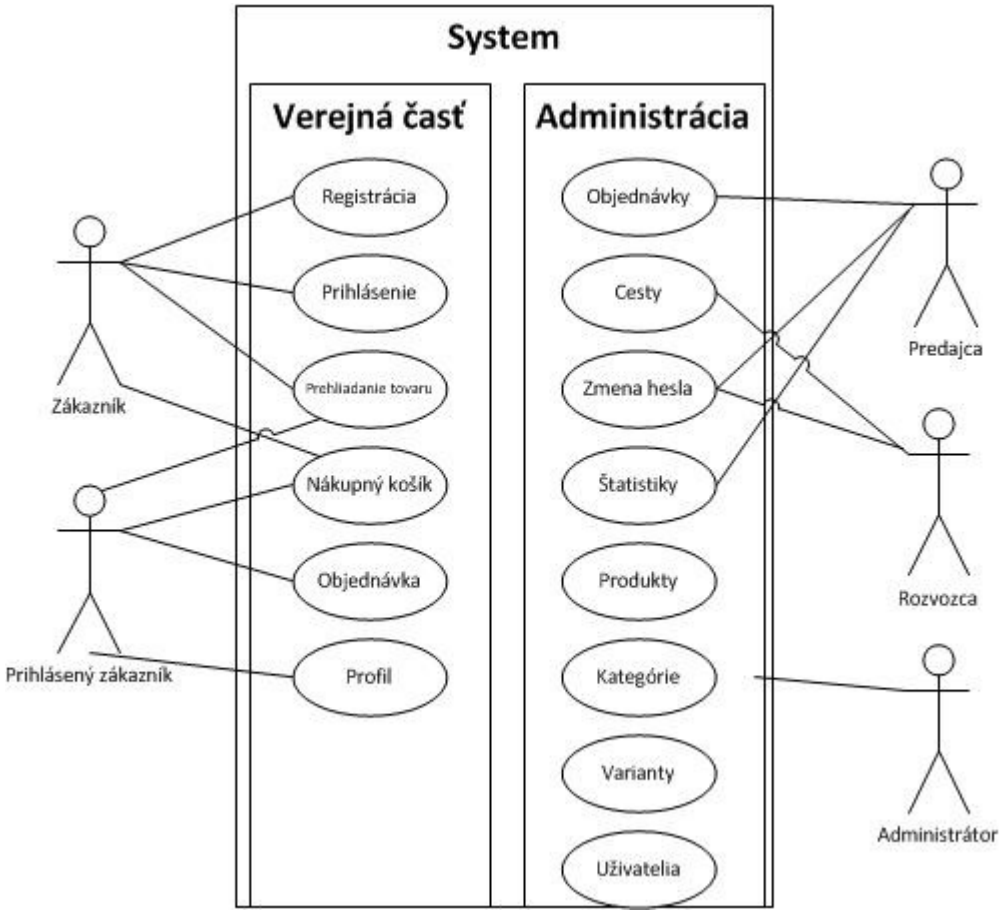
- Status cesty – názov
- Bod cesty – číslo objednávky, adresa, vzdialenosť od predchádzajúceho bodu, čas
- Profil – kontaktné údaje zákazníka (použijú sa pri vypĺňaní objednávok).
- Užívateľ – prihlasovacie meno a heslo, email

5.1.3 Zoznam výstupov

- Zoznam produktov – zoznam produktov rozdelených do kategórií s fotografiou, cenou a popisom.
- Detail produktu - detailný náhľad produktu s popisom, cenou, dostupnými variantami. Detailný náhľad produktu v zákazníckej sekcií bude umožňovať okamžité vloženie produktu do košíka.
- Profil zákazníka – zobrazenie osobných informácií o zákazníkovi
- Zoznam objednávok – zoznam zákazníkových objednávok s dátumom, cenou a statusom. V administračnej sekcií sa budú zobrazovať všetky objednávky
- Detail objednávky – detailný náhľad na kontaktné informácie zákazníka a zoznam položiek objednávky s vybranými variantami a ich cenou, konečná cena objednávky dátum a status objednávky.
- Zoznam užívateľov – zoznam všetkých registrovaných zákazníkov a zamestnancov.
- Detail užívateľa – všetky dostupné údaje o užívateľovi.
- Zoznam ciest – zoznam naplánovaných a uložených ciest, ktorý obsahuje čas a dĺžku cesty, počet bodov ciest, meno zamestnanca ktorý cestu naplánoval a status cesty
- Detail cesty – detailný náhľad na všetky údaje o ceste, všetky body cesty s číslom objednávky a adresou.
- Štatistiky – grafy a tabuľky reprezentujúce a zobrazujúce rôzne štatistické údaje ako napr. počet predaných produktov, počet objednávok, zákazníkov, najazdených kilometrov a.i.

5.1.4 Use Case model

Use Case – model prípadov užitia, znázorňuje popis a použitie funkcií systému z pohľadu rôznych užívateľov. Na obrázku 11 je päť typov užívateľov a funkcie systému ku ktorým majú jednotlivý užívatelia prístup.



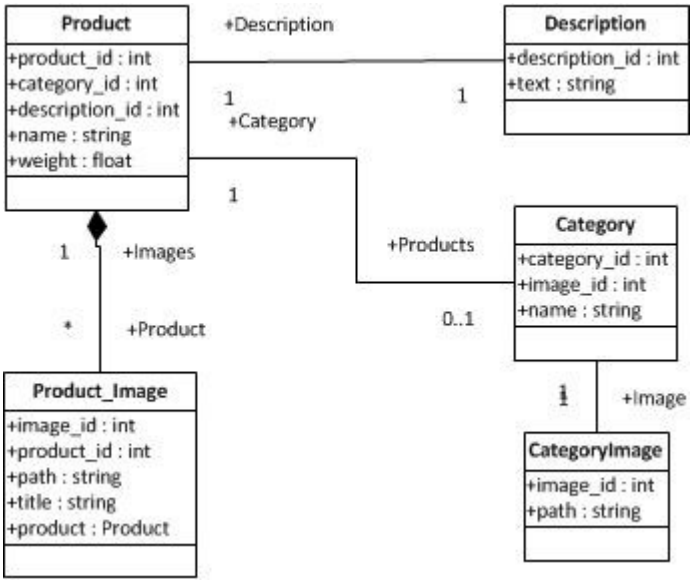
Obr. 11 Use Case model systému

5.2 Konceptuálna analýza

Konceptuálna analýza má za úlohu sémanticky popísať modelovaný systém. Predstavuje platformovo nezávislú dátovú schému navrhovaného systému, ktorá je zbavená všetkých implementačných závislostí. Poskytuje nám teda dostatočný nadhľad tým, že nás zbavuje obmedzení konkrétnej databázovej platformy. Pri konceptuálnom modelovaní sa najčastejšie používa konceptuálny triedny UML model, alebo E-R diagram.

Konceptuálny model UML

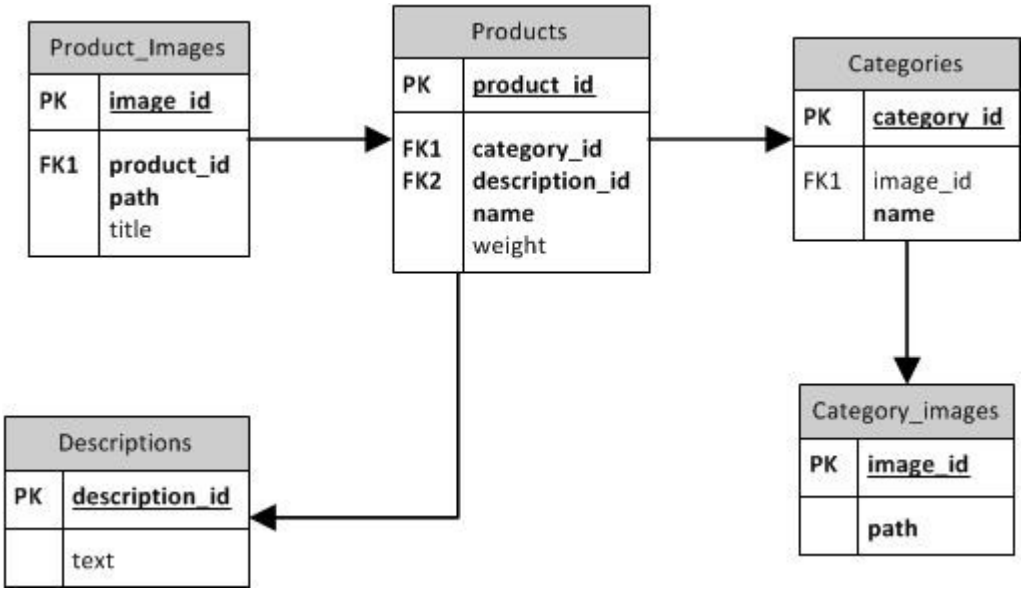
Tento model popisuje statickú štruktúru softwarového systému. Zobrazuje triedy vyskytujúce sa v modelovanom systéme a ich vzájomné vzťahy. Každá trieda je zobrazená ako obdĺžnik so svojim názvom a atribútmi. Okrem samotných atribútov môže každá trieda obsahovať aj metódy resp. operácie, ktoré možno s objektom vykonávať. Najdôležitejšie sú vzťahy medzi triedami, ktoré predstavujú cesty cez ktoré môžu objekty navzájom komunikovať. Tento model je nevyhnutný pre samotnú implementáciu systému. Na obrázku 12 je ukážka triedneho diagramu UML navrhovaného systému. Kompletný diagram je priložený v prílohe B.



Obr. 12 Konceptuálny model UML

Entity-relationship model (ERM)

Tento model popisuje taktiež objekty modelovaného systému ale zameriava sa viac na vývoj databáze. Zobrazuje objekty a relácie medzi nimi pomocou primárnych a cudzích kľúčov. Tvorí dobrý základ pre návrh a vývoj databázy. Na obrázku 13 je ukážka časti navrhovaného systému pomocou ER diagramu. Kompletný ER diagram systému je priložený v prílohe C.



Obr. 13 Entity-relationship model

Oba tieto modely sú dôležité pri ďalšom návrhu a budúcej implementácií systému. Zatiaľ, čo triedny UML diagram kladie pri modelovaní dôraz na návrh aplikácie, ER diagram modeluje štruktúru databázy a vzťahy medzi jej entitami.

Lineárny zápis

Lineárny zápis slúži na prehľadné a stručné zhrnutie všetkých objektov a ich atribútov s ktorými sa bude v systéme pracovať a ktoré sú nevyhnutné pre zabezpečenie požadovanej funkcionality systému.

Nasledujúci zápis stručne popisuje objekty a atribúty navrhovaného systému:

Product(product_id, category_id, description_id, name, weight)

Category(category_id, image_id, name)

CategoryImage(image_id, path)

ProductImage(image_id, product_id, path, title)

Description(description_id, text)

ProductOption(product_option_id, product_id, option_id, name)

ProductOptionValue(product_option_value_id, product_id, option_value_id, product_option_id, price, default)

Option(option_id, name)

OptionValue(option_value_id, option_id, name, image)

Order(order_id, user, firstName, lastName, mail, adress, phone, total, date, comment, order_status_id)

OrderStatus(order_status_id, name)

OrderItem(order_item_id, order_id, product_id, name, price, total, comment)

OrderItemOptionValue(order_item_option_value_id, order_item_id, product_option_id, product_option_value_id, name, value, price)

Route(route_id, duration, distance, owner, date, route_status_id)

RouteStatus(route_status_id, name)

RoutePoint(route_point_id, route_id, order, adress, duration, distance,)

Profile(profile_id, owner, firstName, lastName, adress, news)

5.3 Dátová analýza

Zatiaľ, čo konceptuálny model nám poskytuje implementačne nezávislý pohľad na systém, dátová analýza poskytuje model vyvíjaného systému z pohľadu dátových štruktúr, s ktorými sa bude v systéme pracovať. Detailne popisuje dátové štruktúry a ich obmedzenia, ktoré sú špecifické pre konkrétnu databázovú platformu. Najčastejšie sa na tento účel používajú tzv. dátové slovníky pre každú tabuľku v databáze, ktoré popisujú názvy a tipy atribútov, primárne a cudzie kľúče na iné tabuľky. V tabuľke 3 je ukážka dátového slovníka tabuľky *Orders***.** Kompletný dátový slovník je priložený v prílohe D.

Tabuľka	Orders					
Popis	Tabuľka uchováva všetky objednávky.					
Atribút	**Typ**	**Veľkosť**	**Kľúč**	**Null**	**Index**	**Popis**
OrderId	Int		PK	N	A	Primárny unikátny identifikátor
OrderStatusId	Int		FK	N	N	Odkaz na status resp. stav objednávky
User	Nvarchar	Max	N	A	N	Meno uživateľa, ktorý vytvoril objednávku
Mail	Nvarchar	Max	N	N	N	E-mailová adresa zákazníka
Address	Nvarchar	Max	N	N	N	Adresa zákazníka, resp. doručenia
Phone	Nvarchar	15	N	N	N	Telefónne číslo zákazníka
Status	Nvarchar	20	N	A	N	Meno statusu v ktorom sa objednávka nachádza
Total	Float		N	N	N	Konečná cena objednávky
DateAdded	DateTime		N	N	N	Dátum objednania
FirstName	Nvarchar	30	N	N	N	Meno zákazníka
LastName	Nvarchar	30	N	N	N	Priezvisko zákazníka

Tabuľka 3 Dátový slovník tabuľky Orders

Dátový slovník poskytuje detailnú analýzu jednotlivých tabuliek vrátane ich atribútov, primárnych a cudzích kľúčov, integritných obmedzení a popisom atribútu. V dátovom slovníku sa používajú dátové typy konkrétneho databázového systému.

Podľa dátových slovníkov a ER diagramu, by mal byť programátor schopný vytvoriť databázu vyvíjaného systému.

5.4 Funkčná analýza

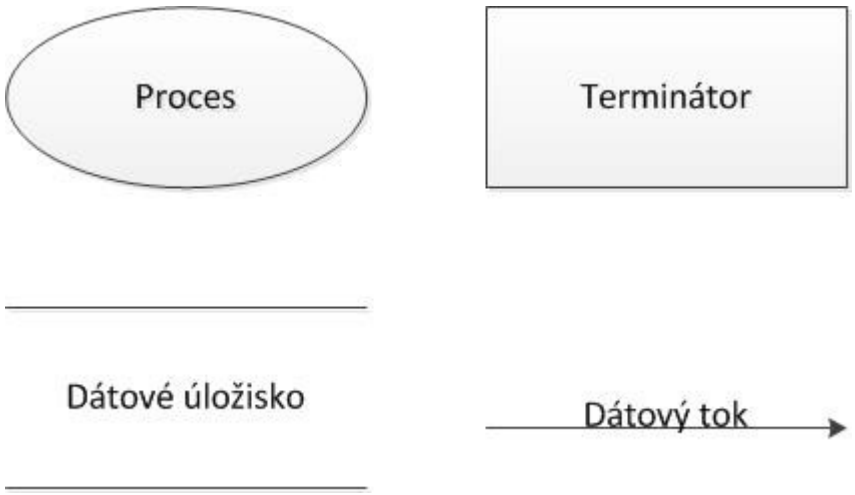
Výstupom predchádzajúcich analýz boli modely zachycujúce systém zo statickej stránky. Úlohou funkčnej analýzy je popísať a modelovať funkcie systému, zachytiť ich interakcie s okolím a formálne popísať algoritmy týchto funkcií. [23]

UML jazyk poskytuje niekoľko nástrojov, pomocou ktorých možno vytvárať tieto modely. Medzi ne patrí napr. diagram dátových tokov, diagram aktivít, stavový diagram apod.

5.4.1 Diagram dátových tokov

Diagram dátových tokov (Data Flow Diagram - DFD), slúži na znázornenie interakcie systému s okolitými elementmi. Zobrazuje dátové toky ktoré prebiehajú medzi systémom a užívateľom, ale aj toky vo vnútri systému.

Jazyk UML [21] používa štyri komponenty, ktoré slúžia na popis tokov v systéme a interakcie medzi nim a okolím. Obrázok 14 ilustruje tieto komponenty.



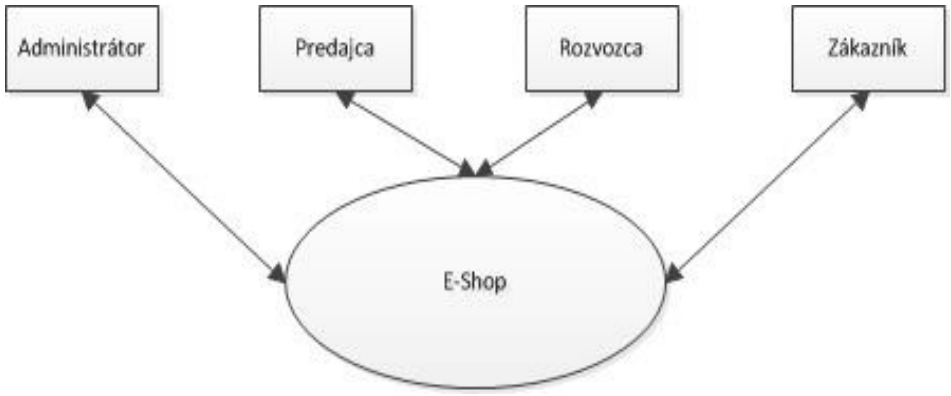
Obr. 14 komponenty Data Flow Diagramu

Proces reprezentuje časť systému alebo funkciu, ktorá premieňa vstupy na výstupy. Terminátor predstavuje externé entity, ktoré komunikujú so systémom. Môžu to byť rôzny aktéri, organizácie alebo iné systémy. Dátové úložisko zabezpečuje perzistentné uloženie údajov. V DFD zvyčajne jedno dátové úložisko pracuje s jedným typom dát ako napr. objednávky, kategórie apod. Dátový tok reprezentuje cestu, po ktorej putujú informácie do systému alebo zo systému (závisí od orientácie šípky)

Aby bol model prehľadnejší, diagramy dátových tokov sa rozdeľujú do tzv úrovní. Najnižšia úroveň popisuje systém ako celok a jeho interakcie s okolitými elementmi. V ďalších úrovniach sa bližšie špecifikujú funkcie a ich interakcie. Čím vyššia je úroveň diagramu, tým sú funkcie a ich interakcie podrobnejšie.

DFD – kontextová úroveň

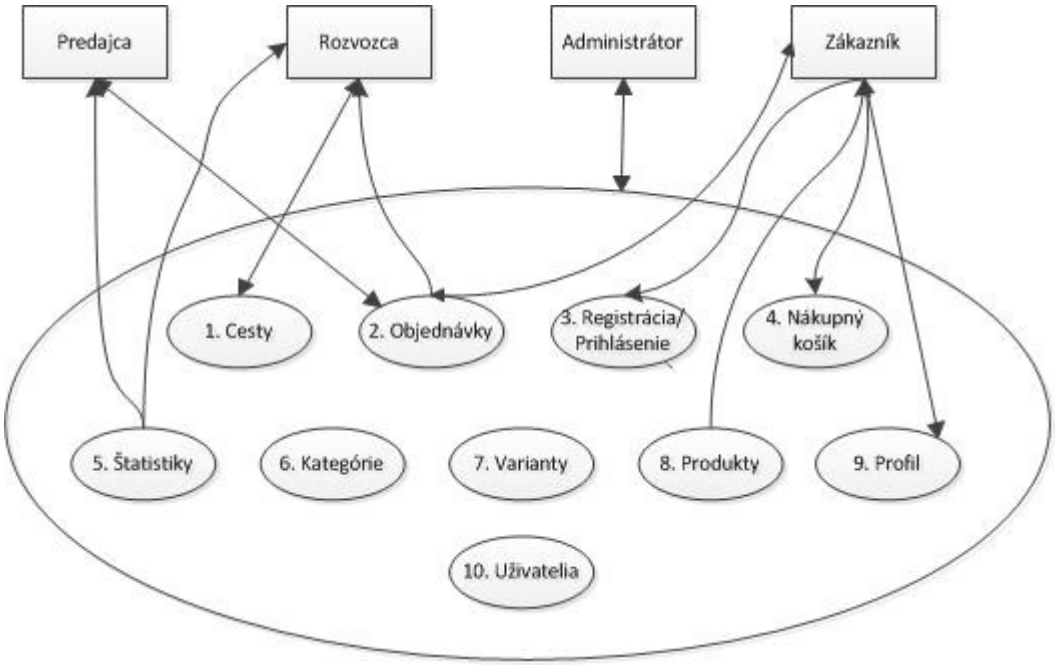
Tento diagram znázorňuje systém ako jeden celok a dátové toky medzi okolitým prostredím. Jediným procesom v diagrame je sám systém a okolité prostredie tvoria zvyčajne užívatelia systému. Na obrázku 15 je kontextový diagram navrhovaného systému.



Obr. 15 Kontextový DFD diagram navrhovaného systému.

DFD - 0. úroveň

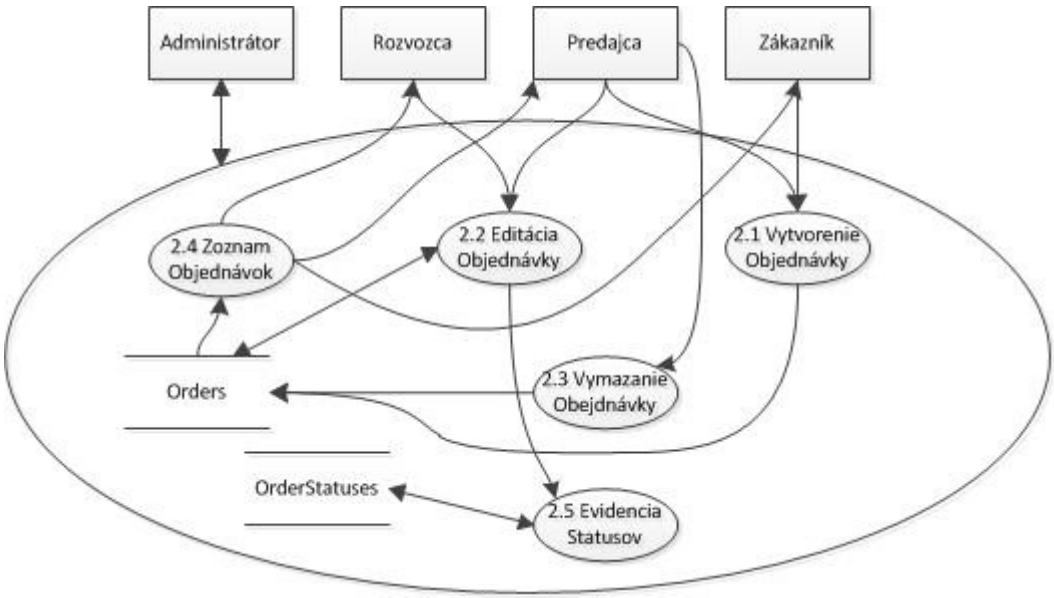
Diagram tejto úrovne rozkladá systém na niekoľko menších subsystémov. Každý tento subsystém zabezpečuje nejakú funkcionalitu a poskytuje služby svojim užívateľom. Znázorňuje dátové toky medzi užívateľmi a týmito subsystémami. Diagram 0.úrovne navrhovaného systému je znázornený na obrázku 16.



Obr. 16 DFD 0. úrovne

DFD – 1. úroveň

Diagram na tejto úrovni podrobnejšie popisuje každý subsystém z úrovne 0. Na tejto úrovni sa pridáva k používaným elementom aj *dátové úložisko* a diagram zobrazuje toky informácií aj v rámci samotného systému. Na obrázku 17 je DFD 1. úrovne navrhovaného systému ktorý opisuje subsystém 2 – *Objednávky*



Obr. 17 DFD 1. úrovne

Ostatné DFD diagramy sú priložené v prílohe E.

5.4.2 Diagram aktivít

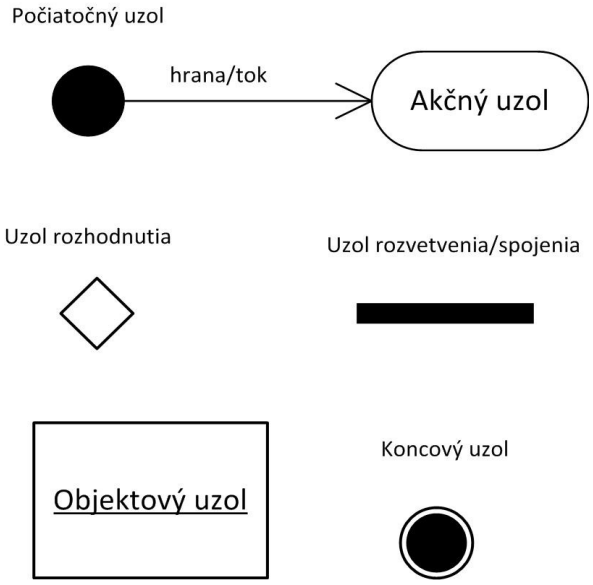
Diagram aktivít (Activity diagram) popisuje chovanie jednotlivých častí systému pomocou sekvencie krokov. Modeluje procesy ako aktivity, ktoré pozostávajú z uzlov (nodes), vzájomne prepojenými hranami (edges).

Ako uvádza [22], existujú tri typy uzlov:

- **Akčné uzly –** reprezentujú samostatnú nedeliteľnú akciu, ktorá by mala premieňať vstupy na výstupy podľa zadanej špecifikácie. Výsledkom takejto akcie môže byť aj nový objekt, alebo sa môže zmeniť stav existujúceho objektu.
- **Riadiace uzly -** majú úlohu riadiť tok medzi jednotlivými akciami. Medzi tieto uzly patria *uzol rozhodnutia, uzol rozvetvenia/spojenia* a *počiatočný/konečný uzol.*
  - *Uzol rozhodnutia* má jednu vstupnú a niekoľko výstupných hrán. Obsahuje taktiež podmienky, ktoré sa navzájom vylučujú a na základe ich splnenia sa vyberá výstupná vetva, ktorou bude tok pokračovať.
  - *Uzol rozvetvenia a spojenia* má úlohu paralelne rozdeliť resp. spojiť tok. Uzol rozvetvenia má jednu vstupnú hranu a viacero výstupných hrán po ktorých pokračuje tok súbežne. Uzol spojenia naopak spája viacero hrán, resp. tokov do jedného výstupného toku.
  - *Počiatočný/koncový uzol* inicializujú počiatočný bod ktorým začína aktivita resp. koncový bod, ktorý aktivitu ukončí.

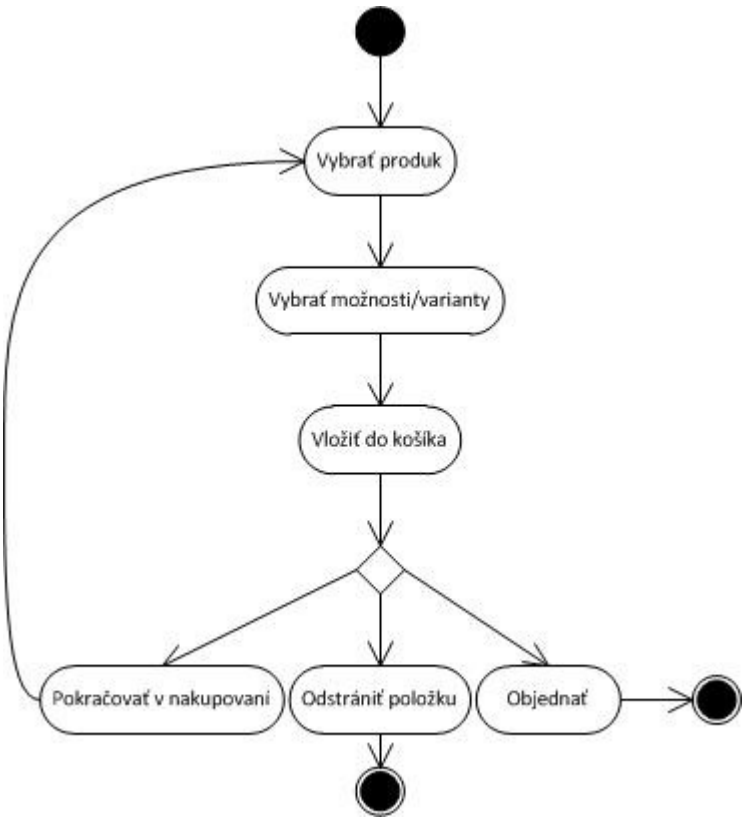
- **Objektové uzly –** reprezentujú objekty, ktoré súvisia s aktivitou alebo sú ňou ovplyvnené.

Na obrázku 18 je zobrazená grafická reprezentácia elementov diagramu aktivít jazyka UML.

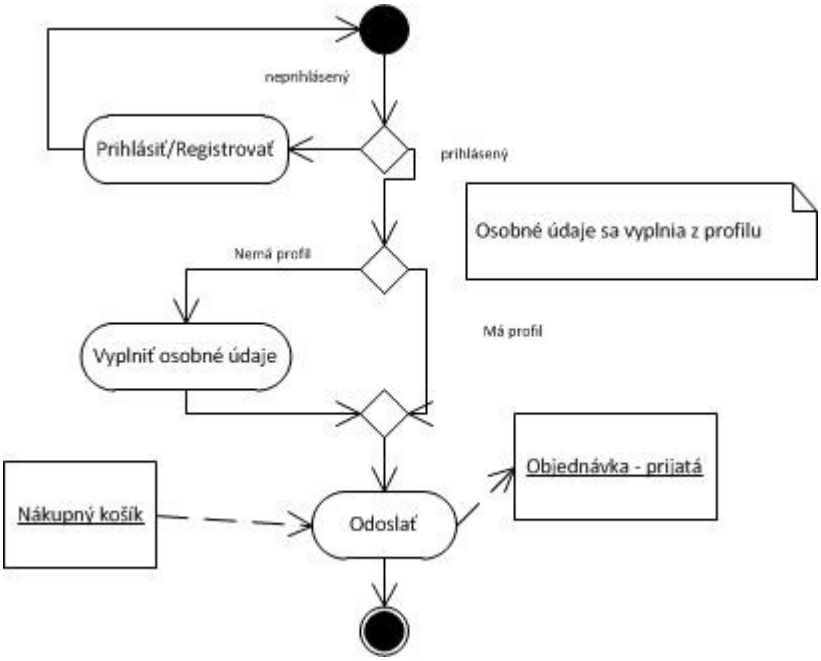


Obr. 18 Elementy Activity diagramu UML

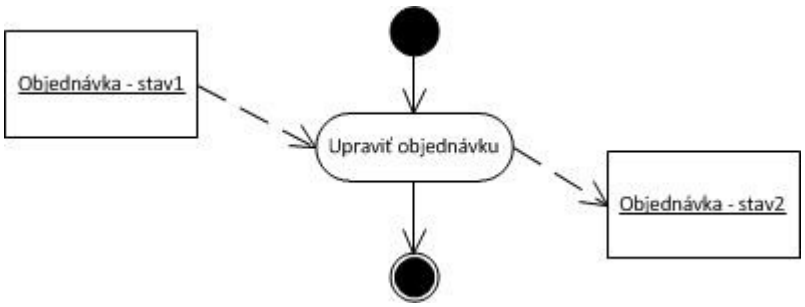
Na nasledujúcich obrázkoch sú niektoré diagramy aktivít navrhovaného systému. Obr. 19 znázorňuje proces vloženia produktu do košíka, obr. 20 proces vytvorenia objednávky, obr. 21 znázorňuje proces úpravy objednávky pri ktorom sa mení jej stav a na obrázku 22 je znázornený proces plánovania ciest a vplyv jednotlivých akcií procesu na stav dotknutých objektov.



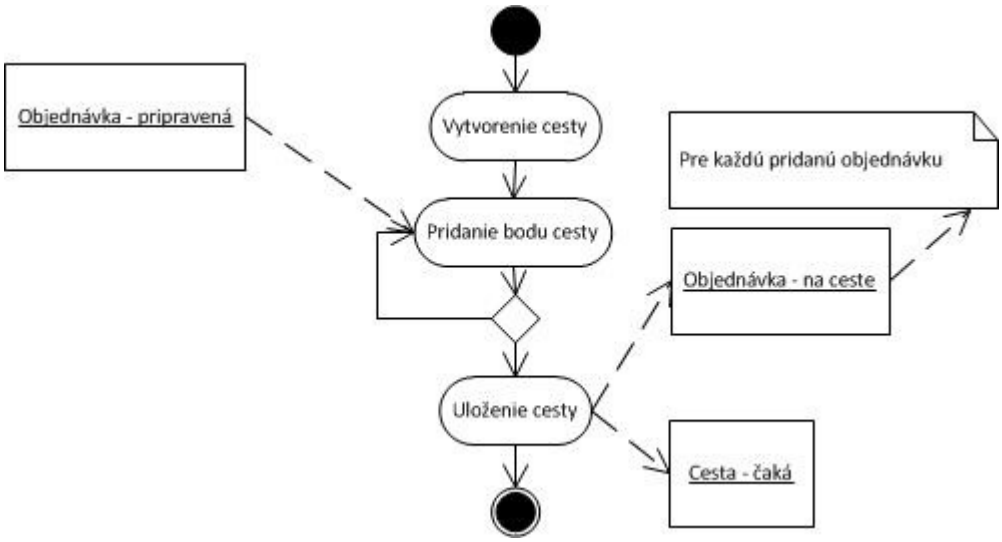
Obr. 19 Activity diagram – vloženie produktu do košíka



Obr. 20 Activity diagram – vytvorenie objednávky



Obr. 21 Activity diagram – úprava objednávky so zmenou jej stavu



Obr. 22 Activity diagram – proces plánovania cesty

Z výstupov predchádzajúcich analýz, ktoré zachytávali ako statickú štruktúru systému tak aj jeho dynamické chovanie, by malo byť možné zostaviť resp. naimplementovať funkčný systém, ktorý bude spĺňať požiadavky definované v špecifikácií zákazníka.

5.5 Návrh implementácie

V tejto kapitole predstavím vlastný návrh implementácie systému. V stručnosti popíšem niektoré použité technológie a postupy a ukážem na príkladoch zaujímavé technologické a programátorské riešenia, ktoré som pri implementácií použil.

5.5.1 Použité technológie a SW

Použité technológie:

- ASP.NET MVC3 (.NETv4)
- Entity framework (v4)
- XHTML 1.0, CSS3
- AJAX
- JavaScript
- LINQ
- Google Maps API

Použitý software:

- Microsoft Visual Studio 2010 – implementácia systému
- Microsoft SQL Server 2008 – databázový server
- Microsoft SQL Management Studio 2010 – nástroj na správu databáze
- Microsoft Visio 2010 – nástroj na tvorbu diagramov UML
- GIMP – grafický procesor na tvorbu grafiky na web

5.5.2 Architektúra systému

Architektúra systému je koncepčný návrh, ktorý určuje štruktúru systému, jeho funkcie a vzťahy medzi nimi a okolím. Vytvára relatívne stabilný rámec riešenia voči okoliu. Dobrý a dostatočne otvorený návrh architektúry by mal umožniť jednoduché zmeny systému. Zvyčajne sa architektúry snažia rozvrstviť a logicky rozdeliť systém. Pri dobrom návrhu je možné systém rozvrstviť aj fyzicky [22]. Najznámejšia je 3-vrstvová architektúra, ktorá od seba oddeľuje prezentačnú, aplikačnú a dátovú vrstvu.

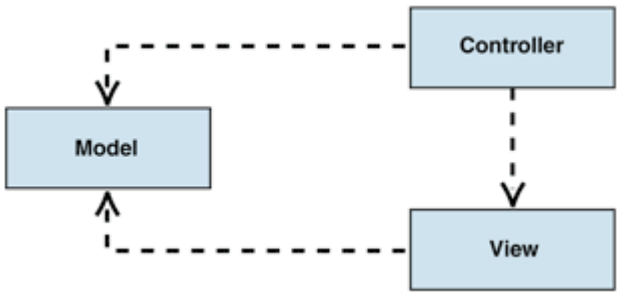
Model-View-Controller

S príchodom a rozšírením webových aplikácií si získala veľkú obľubu architektúra MVC (Model-View-Controller). MVC rozdeľuje aplikáciu na tri komponenty:

- **Model (model)** – obsahuje doménový model aplikácie, ktorý reprezentuje informácie s ktorými sa v systéme pracuje.
- **View (pohľad)** – zobrazuje dáta reprezentované modelom. Zvyčajne sa jedná o HTML stránku.
- **Controller (radič)** – spracováva vstupy od užívateľa a reaguje na ne rôznymi zmenami v modely alebo v pohľade.

Princíp práce v MVC je nasledovný: Užívateľ vykoná nejakú akciu v užívateľskom rozhraní. Zavolá sa radič, ktorý vykoná nejakú operáciu s modelom a vráti užívateľovi nový pohľad ktorý zobrazí informácie reprezentované modelom.

Obrázok 23 ilustruje komponenty a ich vzťahy v architektúre MVC.



Obr. 23 Komponenty architektúry MVC. Zdroj [24]

Zo vzťahov na obrázku 23 vyplýva, že komponenty *View* a *Controller* sú závislé na komponente *Model*. Naopak, *Model* nemá na *Controller* a *View* žiadnu väzbu. Vďaka tejto skutočnosti je komponenta *Model* celkom nezávislá na ostatných komponentách, a je preto veľmi jednoduché robiť zmeny v *Modeli* bez nutnosti výrazného zásahu do ostatných komponent.

Je dôležité spomenúť že samotné MVC nerieši perzistenciu dát modelu. Preto je nutné doplniť *Model* o vrstvu zabezpečujúcu prácu s databázou.

Entity Framework

Aby bol náš systém schopný uchovať dáta rôzneho druhu na dlhšiu dobu, je potrebné integrovať do systému vrstvu, ktorá zabezpečí korektné mapovanie objektov na relačné dáta a manipuláciu s nimi. Nazývame ju tiež vrstva objektovo-relačného mapovania (ORM). Ak sa v systéme nachádza veľký počet objektov, ktoré potrebujeme uložiť v relačnej databáze, bolo by veľmi namáhavé a neefektívne programovať vlastný systém ORM od základov.

V dnešnej dobe je k dispozícií veľa nástrojov, ktoré riešia problematiku ORM. Medzi ne patrí aj Entity Framework (EF), ktorý ponúka rôzne nástroje pre prácu s dátami. Medzi najzaujímavejšie techniky ktoré EF ponúka je samotný proces vytvárania databázy alebo s databázou súvisiaceho modelu.

Prvým spôsobom ako vytvoriť mapovanie medzi databázou a objektmi je tzv. technika *Database First*. Spočíva v tom, že sa najskôr vytvorí schéma databázy, s ktorou sa bude pracovať a pomocou nástrojov EF sa zo schémy vygenerujú modely, na ktoré sa bude databáza mapovať.

Druhým spôsobom je tzv. technika *Code First*, pri ktorej sa najskôr naimplementujú triedy, reprezentujúce potrebné dáta a pomocou nástrojov EF sa na základe ich vygeneruje databáza s odpovedajúcou štruktúrou.

5.5.3 Zabezpečenie systému

Ako vyplýva z požiadaviek systému, systém musí byť schopný pracovať s rôznym typom užívateľov s rôznymi oprávneniami. Musia byť definované pravidlá, na základe ktorých bude možné rozoznať ktorý užívateľ má právo používať určité funkcionality systému. Je potrebné bezpečne uchovať informácie o užívateľoch systému vrátane hesiel a rolí, v ktorých sa užívatelia nachádzajú.

Framework .NET MVC3 ponúka pripravený nástroj pre prácu s užívateľmi, rolami a ich autentifikáciu v systéme. Jedná sa o tzv. Membership provider, ktorý pomocou svojich funkcií a metód umožňuje spravovať užívateľov systému, kontrolovať ich príslušnosť v danej roli, overovať prihlásenie a iné. Heslá sú primárne ukladané ako *hash* kódy, ktoré sú pri autentifikácií užívateľa porovnávané s *hash* kódom zadávaného hesla. Svoje dáta uchováva v databáze, zvyčajne uloženej v zložke App_Data, ale pomocou rôznych nástrojov je možné databázu presunúť na iný server.

Ďalším bezpečnostným prvkom je aj použitie ORM Entity Frameworku, ktorý svojou architektúrou zamedzuje útokom na databázu typu *SqlInject* a pod.

V neposlednom rade je dôležité zabezpečenie aplikácie aj na strane servera. Jednak nastavením správnych oprávnení na priečinky aplikácie, cez aktuálny software na serveri až po pravidelné zálohovanie dát.

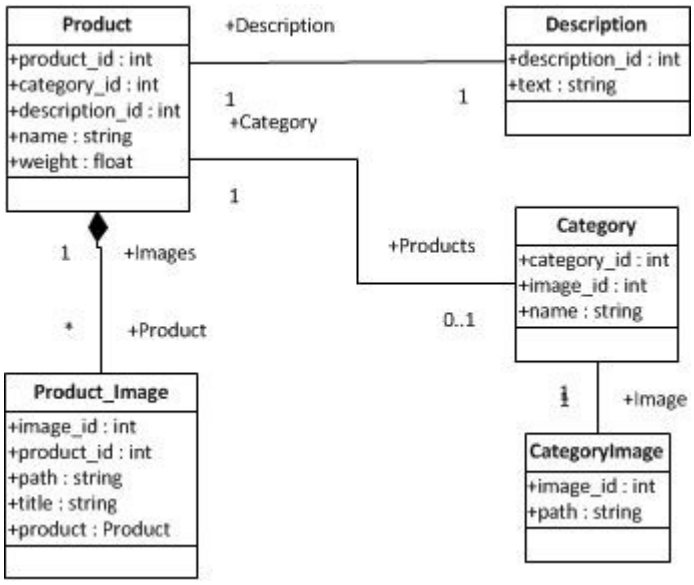
5.5.4 Ukážky implementácie

V nasledujúcej kapitole popíšem niektoré zaujímavé implementačné riešenia s ktorými som sa počas vývoja aplikácie stretol.

MVC a Entity Framework

Ako som spomenul vyššie, ako architektúru systému som sa rozhodol použiť rámec MVC, ktorý prehľadne a logicky oddeľuje jednotlivé časti aplikácie a tým umožňuje jednoduchý a postupný vývoj. S použitím ORM Entity a vývojárskym prístupom Code First, je vývoj veľmi rýchly a jednoduchý.

Prvým krokom pri vývoji aplikácie touto technikou je vytvorenie modelu, ktorý má reprezentovať objekty našej aplikácie. Pri dodržaní určitých zásad pri pomenovávaní atribútov, zabezpečíme správne prepojenie budúcich tabuliek. Na nasledujúcom príklade ukážem ako vytvoriť mapovanie pomocou techniky Code First podľa diagramu na obrázku 24.



Obr. 24 Triedny diagram – Produkt

Najskôr je potrebné naimplementovať všetkých 5 tried podľa diagramu na Obr. 24.

Triedy *Product, ProductImage*, *Category, CategoryImage* a *Description* by vyzerali nasledovne:

```
public class Product {
      public int ProductId {get;set;}
      public int CategoryId {get;set;}
      public virtual Category Category {get;set;}
      public int DescriptionId {get;set;}
      public virtual Description Description {get;set;}
      public string Name {get;set;}
      public float weight {get;set;}
      public virtual IList<ProductImage> Images {get;set;}
}
```

```
public class ProductImage {
      public int ProductImageId {get;set;}
      public string Url {get;set;}
      public string Title {get;set;}
      public int ProductId {get;set;}
      public virtual Product Product {get;set;}
}
public class Category {
      public int CategoryId {get;set;}
      public string Name {get;set;}
      public int CategoryImageId {get;set;}
      public virtual CategoryImage Image {get;set;}
}
public class CategoryImage {
      public int CategoryImageId {get;set;}
      public string Url {get;set;}
}
public class Description {
      public int DescriptionId {get;set;}
      public string Text {get;set;}
}
```

Zo zdrojového kódu si môžeme všimnúť, že je potrebné dodržiavať určité zásady pri pomenovávaní premenných. Ak chceme aby bol atribút primárnym kľúčom, musíme ho pomenovať rovnako ako názov triedy a pridať na koniec názvu reťazec *Id.* Napr. v triede *Description* bude primárnym kľúčom atribút *DescriptionId*. Druhou možnosťou je použiť pred atribútom anotáciu *[key]*, ktorá zabezpečí, že atribút bude primárnym kľúčom aj napriek inému názvu. Cudzie kľúče sa označujú obdobne. Zaujímavosťou sú atribúty s kľúčovým slovom *virtual*. Pomocou týchto atribútov Entity Framework namapuje zodpovedajúce závislé objekty.

Aby Entity framework vedel, ktoré triedy z modelu má použiť pri vytváraní databáze a následnej práce s ňou, musíme vytvoriť triedu ktorá dedí zo špeciálnej triedy *DbContext* a pridať do nej záznamy typu *DbSet<T>* pre každú triedu s ktorou chceme pracovať.

Nasledujúci kód ukazuje triedu *ShopContext*, ktorá zabezpečí prvotnú inicializáciu databázy a následnú prácu s entitami *Product, ProductImage, Description, Category* a *CategoryImage*:

```
public class ShopContext : DbContext {
      public DbSet<Product> Products {get;set;}
      public DbSet<ProductImage> ProductsImages {get;set;}
      public DbSet<Description> Descriptions {get;set;}
      public DbSet<Category> Categories {get;set;}
      public DbSet<CategoryImage> CategoryImages {get;set;}
}
```

Pri prvom inštancovaní objektu *ShopContext,* Entity Framework vytvorí štruktúru databázy na základe pripravených tried. Tabuľky databázy, ktoré sú frameworkom vygenerované, obsahujú rovnaké atribúty ako triedy, ktoré sú prevedené na dátové typy cieľovej databázy s odpovedajúcimi primárnymi a cudzími kľúčmi.

Užitočným nástrojom pre dotazovanie nad kolekciami a databázami je LINQ – Language Integrated Query. Je to integrovaný dotazovací jazyk, ktorý umožňuje dotazovanie nad rôznymi typmi dát.

- LINQ to Objects – umožňuje dotazovanie nad dátami uloženými v pamäti. Jedná sa o rôzne polia, zoznamy, kolekcie a iné triedy implementujúce rozhranie *Ienumerable<T>*.
- LINQ to SQL – umožňuje dotazovanie nad databázami. Pri takomto dotazovaní sa nevykonáva dotaz pomocou dotazovacieho enginu LINQ ale príkazy sa mapujú na príkazy SQL a vykonajú sa na strane databázy.
- LINQ to XML – umožňuje dotazovanie nad XML súbormi plne objektovým prístupom.

Na nasledujúcom kóde je ukážka využitia LINQ dotazovania v kombinácií s Entity Frameworkom:

```
private ShopContext db = new ShopContext();
var products = db.Products; // všetky produkty
// produkty lacnejšie ako 100 korún
var cheapProducts = db.Products.Where(p => p.price<100);
```

Nad dátami môžeme použiť a skombinovať rôzne funkcie známe aj z databáz ako napr. *Count(), Sum*(), *OrderBy()* ale aj iné funkcie ako napr. *Find*(), *Ski*p() apod.

Google Maps

Jednou z dôležitých funkcií vyvíjaného systému je plánovanie ciest ku zákazníkom. Každý bod cesty predstavuje jedna objednávka, v ktorej je dôležitá najmä adresa. Výsledkom cesty by mala byť čo najkratšia trasa medzi týmito bodmi. Pre reprezentáciu výsledku je potrebné integrovať do systému nejakú mapu.

V mojom riešení som sa rozhodol použiť službu *Google Maps*, ktorá ponúka širokú škálu nástrojov na prácu s mapami. Okrem samotného zobrazovania máp, ponúka funkcie na konverziu adries na geo-súradnice, počítanie trás medzi rôznymi bodmi apod.

Najdôležitejšou službou pre náš systém je *Google Maps Directions API*. Je to rozhranie pre aplikácie, ktoré umožňuje počítanie ciest zo zadaných bodov a podľa definovaných parametrov. Je možné vybrať si medzi dvoma druhmi API. Prvé rozhranie funguje na základe zasielania špeciálnych požiadaviek na API server, ktorý vracia ako odpoveď dáta vo formáte JSON alebo XML. Druhé rozhranie využíva jazyk JavaScript a pracuje sa s ním pomocou objektov.

V mojom systéme som sa rozhodol použiť API založené na JavaScripte. V nasledujúcej ukážke je zdrojový kód, ktorý inicializuje mapu na stránke a zobrazí cestu medzi Ostravou a Brnom.

```
<head>
 <script type=”text/javascript  src=”http://maps.google.com/maps/api/js”>
</head>
<body>
 <div id=”map”></div>
 <script type=”text/javascript>
  var directionService= new google.maps.DirectionService();
  var directionDisplay= new google.maps.DirectionRender();
  var map = new google.maps.Map(document.GetElementById(“map”)),
    {zoom:7,
     mapTypeId: google.maps.MapTypeId.ROADMAP});
  directionDisplay.setMap(map);
  var request = {
    origin: 'Ostrava',
    destination: 'Brno',
```

```
    travelMode: google.maps.DirectionsTravelMode.DRIVING};
  directionService.route(request, function(response, status) {
   if (status==google.maps.DirectionsStatus.OK) {
    directionDisplay.setDirection(response);}
   });
 </script>
</body>
```

DirectionsService je služba, ktorá zabezpečuje spracovanie požiadavku a vrátenie objektu s vypočítanou cestou. *DirectionRender* zobrazuje výsledok z na mape. Samotnú mapu predstavuje objekt *Map*, ktorý zobrazí mapu v sekcií *div* s identifikátorom *map.* Funkcia *directionsDisplay.setMap()* nastaví renderu mapu na ktorej má zobraziť výsledok. Nasleduje zostavenie samotnej požiadavky *request,* kde v tomto prípade nastavujem adresu štartu a adresu cieľa. *DirectionService* spracuje požiadaviek a vráti výsledok v podobe JSON objektu. Tento objekt je predaný funkcií *DirectionsRender*, ktorý obdržaný výsledok zobrazí na mape. Pri ukladaní výsledku do databázy využívam taktiež tento objekt, z ktorého si zmapujem objekty *Route* a *RoutePoints*.

Je potrebné uviesť, že v mojej práci používam voľnú licenciu Google Maps Directions API [19], ktorá obmedzuje používanie výsledkov tejto služby výhradne na ich zobrazovanie na mape. Pre iné použitie je potrebné zakúpiť tzv. Business licenciu.

Zabezpečenie

Aby bolo možné zabezpečiť prístup k jednotlivým funkciám systému na základe oprávnenia užívateľa, je potrebné definovať pravidlá na základe ktorých je možné rozhodnúť, či má užívateľ na prístup k funkcií právo, alebo nie. Ako som spomenul v predchádzajúcej kapitole, jadrom zabezpečenia vyvíjaného systému bude komponenta frameworku .NET, *Membership Provider*, ktorá zabezpečuje prihlasovanie a autentifikáciu užívateľov.

Keďže všetky vstupy ktoré prichádzajú od užívateľa do systému v MVC spracováva controller, je tým najlepším miestom kde definovať pravidlá pre prístup k jeho akciám. Na definovanie pravidla sa používajú anotácie *[Authorize]*, ktoré môžu mať rôzne parametre. Na nasledujúcom kóde je znázornené zabezpečenie controlleru *ProfilesController*.

```
[Authorize(Roles="Customer")]
public class ProfileController {
}
```

Anotácia *[Authorize]* označuje, že ku controlleru má prístup len prihlásený užívateľ. Parameter *Roles="Customer"* ďalej upresňuje, že užívateľ musí byť v roli *Customer.*

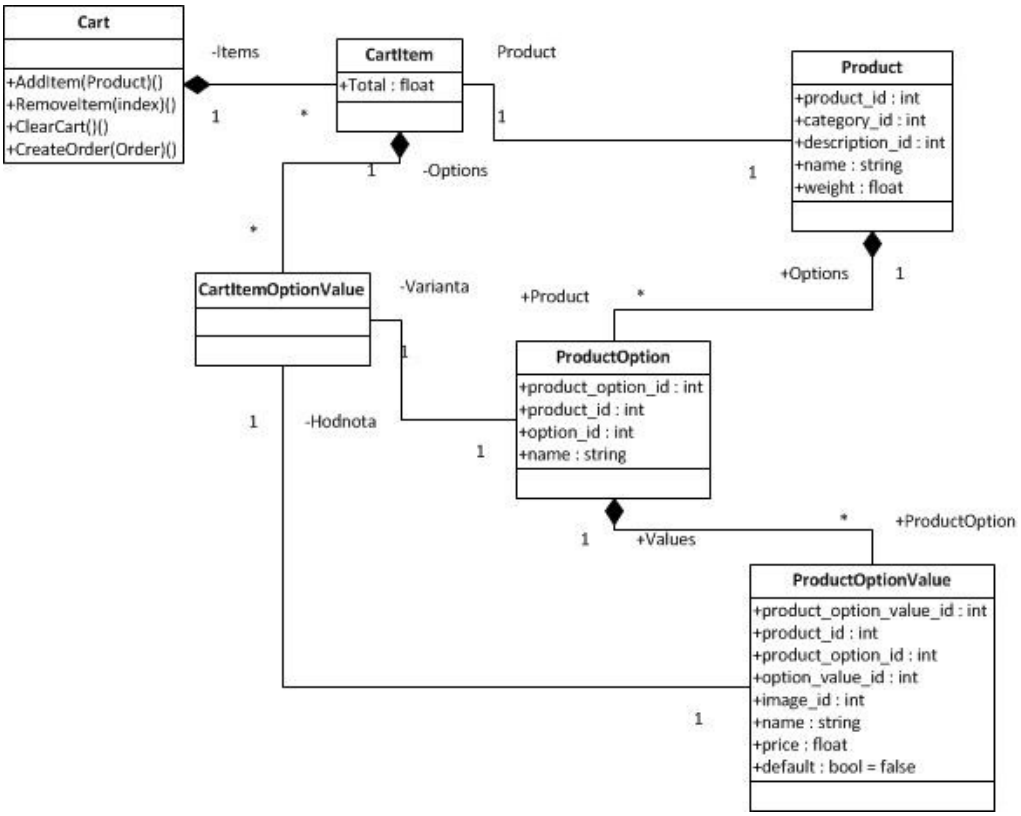
Anotácie je možné použiť aj na samotné akcie controlleru a tým sprístupniť jednotlivé akcie rôznym užívateľom.

Využívať funkcie *Membership Provider* je možné aj vo View aj v Controller. Najčastejšie som využíval funkcie ako *User.Identity.Name*, ktorá vracia meno aktuálne prihláseného užívateľa, alebo funkciu *User.IsInRole()*, ktorá skontroluje, či je aktuálny užívateľ v danej roli.

Nákupný košík

Nákupný košík je neoddeliteľnou súčasťou moderného a pohodlného nakupovania cez internet ale aj v obchode. Jeho primárnou funkciou je umožniť užívateľovi, odložiť si vybraný tovar do ukončenia nákupu, v našom prípade vytvorenia objednávky. Nákupný košík by mal užívateľa sprevádzať po celú dobu nákupu až po vytvorenie objednávky.

Keďže náš systém umožňuje voľby rôznych variant produktu, je potrebné uchovať pri každej položke v košíku aj vybrané varianty. Na obrázku 25 je znázornený triedny diagram nákupného košíka.



Obr. 25 UML triedny diagram nákupného košíka

Nákupný košík nie je potrebné ukladať do databázy. Jeho stav je uložený v *Session* užívateľa na strane servera.

Session umožňuje uloženie nejakej hodnoty (reťazec, číslo, objekt, apod.), ktorá je k dispozícií počas celého prístupu užívateľa na server. Všetky informácie sú uložené na strane servera v operačnej pamäti. Na strane klienta je uložený iba identifikátor aktuálnej *session*, typicky ako *cookie*, podľa ktorého sa identifikuje užívateľ na strane servera. Každá *session* má určitú dobu platnosti (defaultne 20 minút), po ktorej je zo strany servera ukončená.

5.6 Testovanie a nasadenie

Pred nasadením systému do ostrej prevádzky je nutné otestovať jeho správnosť a funkčnosť. Testovanie je súbor procesov, ktoré slúžia na kontrolu kvality softvéru. Software sa testuje z hľadiska funkčnosti, spoľahlivosti, výkonnosti a pod.

Prvá fáza testovania prebieha už počas fázy programovania, samotným programátorom, ktorý postupne overuje funkčnosť a správnosť systému ktorý vyvíja. Druhá fáza prebieha pri odovzdávaní produktu zákazníkovi, kedy sa overí jeho funkčnosť a zhoda zo zadaním. Niektoré situácie, ktoré môžu vzniknúť počas ostrej prevádzky systému, sa nemusí podariť zachytiť vo fáze testovania. Preto je dôležité overovanie funkčnosti a správnosti systému aj počas prevádzky.

Nasadenie systému pre produkčné použitie sa skladá z nasledujúcich krokov:

- Zabezpečenie vhodného webhostingu s podporou požadovaných technológii
- Inštalácia systému podľa inštalačného manuálu, viz príloha F.
- Naplnenie obsahu databáze (kategórie, produkty, obrázky apod.)
- Spustenie ostrej prevádzky
- Registrácia stránok vo vyhľadávačoch a webových katalógoch

Systém bude pre účely testovania dostupný z adresy koxo.somee.com s nasledujúcimi prístupovými údajmi pre administrátora: meno: *Administrator*, heslo: *administrator*

6. Záver

Internetové obchodovanie je čoraz populárnejšou formou predaja. Umožňuje obchodníkovi rapídne znížiť náklady a zároveň zvýšiť počet potencionálnych zákazníkov. Pre samotného zákazníka je veľkou výhodou, možnosť nakupovania z pohodlia domova a často aj ponuka nižších cien oproti kamenným obchodom. Nižšie ceny úzko súvisia práve so znížením nákladov zo strany predajca.

V mojej práci som mal za ciel vyvinúť systém pre podporu elektronického obchodovania, so zameraním na predaj pizze. Unikátnosť riešenia zabezpečuje systém pre plánovanie ciest ku zákazníkom. Pri spracovaní témy som si osvojil základné teoretické pojmy z oblasti informačných systémov a metodík vývoja softwaru. Pri analýze a návrhu riešenia som prešiel všetkými fázami analýzy a návrhu systému, a osvojil som si vyjadrovacie prostriedky pre ich zápis. Pri implementácií som využil celú radu moderných technológií ako ASP.NET MVC, Entity Framework, XHTML, CSS, AJAX, LINQ, Google Maps apod.

Jedným z možných vylepšení projektu do budúcnosti sú otváracie hodiny. Najmä ak bude systém využívaný pizzériou, ktorá prijíma a spracúva objednávky len v určitom čase, bolo by vhodné obmedziť možnosť nakupovania a objednávania len na čas otváracích hodín.

Druhým vylepšením, ktoré sa ponúka, je nejaký systém zliav a akcií, ktoré by mohol spravovať predajca. Buď by sa jednalo o množstvové zľavy ako napr. „Pri objednávke troch produktov, štvrtý zdarma“ a podobne, alebo o časovo obmedzené zľavy na určitý produkt alebo kategóriu.

Keďže systém podporuje len registrovaných užívateľov, bolo by možné integrovať do systému nejaký bodový, resp. kupónový systém. Jednalo by sa o rôzne odmeny za nákupy vo forme bodov, ktoré by zákazníci mohli využiť ako zľavu pri budúcich nákupoch.

Ďalším vylepšením, ktoré by zvýšilo efektivitu a pohodlie práce so systémom je väčšia integrácia Javasriptu a AJAXu, ktorá by umožnila interaktívnejšiu a rýchlejšiu prácu s užívateľským prostredím. Ide hlavne o dynamické pridávanie položiek vo formulároch, interaktívnejšiu prácu s obrázkami apod.

7. Zoznam použitej literatúry

[1] HARDCASTLE, Elizabeth. Business information systems. 2008. ISBN 978-87-7681-463-2.

[2] BRUCKNER, Tomáš. Tvorba informačných systémů: principy, metodiky, architektury. 1. vyd. Praha: Grada, 2012, 357 s. Management v informační společnosti. ISBN 978-80-247-4153-6.

[3] Common Types of Information Systems. [online]. [cit. 2013-05-05]. Dostupné z: http://eternalsunshineoftheismind.wordpress.com/2013/02/19/common-types-of-information-systems/

[4] Types of information systems. [online]. [cit. 2013-05-05]. Dostupné z: http://bisom.uncc.edu/courses/info2130/Topics/istypes.htm

[5] KASSE, Tim. Practical insight into CMMI [online]. Boston: Artech House, c2004 [cit. 2013-05-05]. Dostupné z: www.cmmi.de/#el=CMMI-DEV/0/HEAD/folder/folder.CMMI-DEV

[6] BUCHALCEVOVÁ, Alena. Metodiky vývoje a údržby informačních systémů: kategorizace, agilní metodiky, vzory pro návrh metodiky. 1. vyd. Praha: Grada, 2005, 163 s. ISBN 80-247-1075-7.

[7] Software development life cycle. [online]. [cit. 2013-05-05]. Dostupné z: http://www.utahta.wikispaces.net/Software+Development+Life+Cycle

[8] RUP workflows. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2013-05-05]. Dostupné z: http://en.wikipedia.org/wiki/File:RUP_Workflows2.gif

[9] MCCONNELL, Steve. Dokonalý kód: umění programování a techniky tvorby software. Vyd. 1. Brno: Computer Press, 2005. ISBN 80-251-0849-X.

[10] Introduction to the Microsoft Solutions Framework. Technet [online]. [cit. 2013-05-05]. Dostupné z: http://technet.microsoft.com/en-us/library/bb497060.aspx

[11] Magento Community Edition. [online]. [cit. 2013-05-05]. Dostupné z: http://www.magentocommerce.com/download?icid=topnav

[12] NopCommerce. Features [online]. [cit. 2013-05-05]. Dostupné z: http://www.nopcommerce.com/featurelist.aspx

[13] Konakart. Features [online]. [cit. 2013-05-05]. Dostupné z: http://www.konakart.com/product/features/shopping-experience

[14] Zen-cart. Features list [online]. [cit. 2013-05-05]. Dostupné z: http://www.zen-cart.com/wiki/index.php/Features_List

[15] http://www.adaptivesd.com/articles/messy.htm

[16] SCRUM. Certicon [online]. [cit. 2013-05-05]. Dostupné z: http://www.certiconglobal.com/content/scrum

[17] KNIBERG, Written by Henrik. Scrum and xp from the trenches: how we do scrum. [S.l: C4Media Inc.], 2007. ISBN 978-143-0322-641.

[18] Eshop-rychlo [online]. 2011 [cit. 2013-05-05]. Dostupné z: Eshop-rychlo.sk

[19] Webaster [online]. 2011 [cit. 2013-05-05]. Dostupné z: http://webaster.sk/

[20] Fast-centrik [online]. 2011 [cit. 2013-05-05]. Dostupné z: http://www.netdirect.cz/fastcentrik/pronajem-e-shopu/

[21] ARLOW, Jim a Ila NEUSTADT. UML 2 a unifikovaný proces vývoje aplikací: objektově orientovaná analýza a návrh prakticky. Vyd. 1. Překlad Bogdan Kiszka. Brno: Computer Press, 2007, 567 s. ISBN 978-80-251-1503-9.

[22] FOWLER, Martin. UML distilled: a brief guide to the standard object modeling language. 3rd ed. Boston: Addison-Wesley, c2004, xxx, 175 p. ISBN 03-211-9368-7.

[23] ŘEPA, Václav. Analýza a návrh informačních systémů: a brief guide to the standard object modeling language. 1.vyd. Praha: Ekopress, 1999, 403 s. ISBN 80-861-1913-0.

[24] MSDN. Model-View-Controller [online]. [cit. 2013-05-05]. Dostupné z: http://msdn.microsoft.com/en-us/library/ff649643.aspx

[25] Google Maps API Web Services. The Google Directions API [online]. [cit. 2013-05-05]. Dostupné z: https://developers.google.com/maps/documentation/directions